Latent variable decoding in biological and artificial agents

Towards a unified approach

Pietro Vertechi

Dissertation presented to obtain the Ph.D degree in Neuroscience

Instituto de Tecnologia Química e Biológica António Xavier | Universidade Nova de Lisboa

Oeiras,
September, 2020

UNIVERSIDADE NOVA DE LISBOA

Latent variable decoding in biological and artificial agents

Towards a unified approach

Pietro Vertechi

Dissertation presented to obtain
the Ph.D degree in Neuroscience

Instituto de Tecnologia Química e Biológica António Xavier
Universidade Nova de Lisboa

Research work coordinated by:

Champalimaud
Foundation

Oeiras, September, 2020

Latent variable decoding in biological and artificial agents: towards a unified approach.

Pietro Vertechi

A Dissertation
Presented to the Faculty
of Universidade Nova de Lisboa
in Candidacy for the Degree
of Doctor of Philosophy

Supervisor: Zachary F. Mainen

September, 2020

Never stay up on the barren heights of cleverness, but come down into the green valleys of silliness.

Ludwig Wittgenstein

Acknowledgments

There are several people without whom this work would not have been possible. I am indebted to Christian K. Machens, who introduced me to theoretical neuroscience and taught me many of the skills required in academia. Even though I was unable to express this at the time, I highly value his focus on clarity in thought and writing. I am grateful to Wieland Brendel for the many interesting conversations we had over time, and for teaching me how to program.

I thank Eran Lottem and Dario Sarra for their invaluable guidance and support: I would have felt lost without them. My thanks also go to the labmates with whom I collaborated more closely in the main project of my Ph.D.: Beatriz Godinho, Tiago Quendera, Isaac Treves, and Matthijs Nicolai Oude Lohuis.

I am thankful to Mattia G. Bergomi for showing me a different way of doing research, and for supporting me during a difficult phase of my Ph.D. I also thank our collaborators: Massimo Ferri and Patrizio Frosini.

I wish to express my gratitude to my supervisor Zachary F. Mainen. Thanks to his focus on independence and freedom, as well as his ability to see value in seemingly outlandish projects, I was able to attempt many different ideas during my Ph.D. and become a more independent researcher.

I am thankful for the moral and emotional support of my family and my partner over the years. I am grateful to Madalena Fonseca and Cindy Poo for their advice on professional and psychological difficulties.

Finally, I acknowledge the financial support generously granted by the *Fundação para a Ciência e a Tecnologia* and by the *Fundação Champalimaud*.

Abstract

Decision-making in the presence of uncertainty is a pervasive computation. Latent variable decoding—inferring hidden causes underlying visible effects—is commonly observed in nature, and it is an unsolved challenge in modern machine learning.

On many occasions, animals need to base their choices on uncertain evidence; for instance, when deciding whether to approach or avoid an obfuscated visual stimulus that could be either a prey or a predator. Yet, their strategies are, in general, poorly understood.

In simple cases, these problems admit an optimal, explicit solution. However, in more complex real-life scenarios, it is difficult to determine the best possible behavior. The most common approach in modern machine learning relies on artificial neural networks—black boxes that map each input to an output. This input-output mapping depends on a large number of parameters, the weights of the synaptic connections, which are optimized during learning.

While useful in practice, these solutions are not *intelligible*: the specific parameter values reached at the end of learning do not shed light on the nature of the problem. Moreover, even when artificial neural networks can be trained to almost optimally decode a latent variable from a series of noisy observations, the link with biological agents and biological neural networks remains unclear.

Here, we approach the latent variable decoding problem using three distinct lenses: animal behavior, intelligible machine learning, and modeling of biological neural networks.

First, we examine animal behavior in carefully designed tasks, to understand whether mice and humans can learn what variables are relevant in a task, and how those variables should be first inferred, and then used, to solve the task successfully. Using optogenetics, we distinguish the role of different prefrontal cortical areas in performing this computation.

Then, we turn to a normative approach. Even though optimal solutions are often intractable, we develop different flavors of *parametric machines*: simple, but computationally flexible, artificial agents. We implement these machines in *PyTorch* and showcase their performance on nonlinear regression and image classification tasks on small datasets.

Finally, investigating which synaptic plasticity mechanisms would correspond to optimal decoding, we build a bridge between normative and biological neural solutions. We find that efficient coding can be achieved via a class of simple plasticity rules that aim to tighten the excitation/inhibition balance of the network.

Taken together, these complementary approaches provide an initial step towards a unified normative understanding of decision-making under uncertainty in biological and artificial agents.

Título e Resumo

Descodificação de variáveis latentes em agentes biológicos e artificiais: em direção a uma estratégia comum.

Tomar decisões na presença de incerteza é uma computação omnipresente. A descodificação de variáveis latentes—inferir causas escondidas subjacentes a efeitos visíveis—é frequentemente observada na natureza, e é também um desafio não resolvido no setor de aprendizagem de máquina.

Em muitas ocasiões, os animais precisam de basear as suas escolhas em evidências incertas, por exemplo quando decidem se se devem aproximar ou afastar dum estímulo visual ofuscado que poderia ser uma presa ou um predador. No entanto, as suas estratégias são, no geral, pouco compreendidas.

Em casos simples, estes problemas admitem uma solução ótima e explicita. Todavia, em casos realísticos mais complexos, é difícil determinar o melhor comportamento possível. A abordagem mais comum em aprendizagem de máquina baseia-se no conceito de rede neuronal artificial: uma "caixa negra" que associa a cada *input* um *output*. Esta associação entre *input* e *output* depende dum grande número de parâmetros que são otimizados durante a aprendizagem.

Embora úteis na prática, estas soluções não são *inteligíveis*: os valores dos parâmetros obtidos no fim da aprendizagem não esclarecem a natureza do problema. Além disso, mesmo quando redes neuronais artificiais podem ser treinadas para descodificar quase otimamente uma variável latente a partir duma série de observações, a relação com agentes e redes neuronais biológicas permanece pouco clara.

Aqui, abordamos o problema de descodificação de variáveis latentes sob três óticas distintas: comportamento animal, aprendizagem de máquina inteligível e modelação de redes neuronais biológicas.

Primeiro, examinamos o comportamento animal em tarefas cuidadosamente concebidas, para perceber se os ratinhos e os humanos podem aprender quais variáveis são relevantes numa tarefa, e como essas variáveis devem ser primeiro inferidas, e depois utilizadas, para resolver a tarefa com sucesso. Usando optogenética, distinguimos o papel de diferentes áreas do córtex pré-frontal na realização deste cômputo.

Depois, recorremos a uma estratégia normativa. Ainda que as soluções ótimas sejam frequentemente espinhosas, desenvolvemos diferentes tipos de *máquinas paramétricas*: agentes artificiais simples mas computacionalmente flexíveis. Implementamos estas máquinas em *PyTorch* e demonstramos o seu desempenho em tarefas de regressão não linear e classificação de imagens em pequenos conjuntos de dados.

Finalmente, tentamos construir uma ponte entre os agentes normativos e as soluções biológicas neuronais, investigando quais mecanismos plausíveis de plasticidade sináptica correspondem a uma descodificação ótima. Descobrimos que a codificação eficiente pode ser alcançada através duma classe de regras de plasticidade simples que visam a melhorar o equilíbrio de excitação/inibição da rede.

No seu conjunto, estas estratégias complementares representam um primeiro passo na direção duma compreensão unificada do processo de tomada de decisões sob incerteza em agentes biológicos e artificiais.

Author Contributions

This thesis was written by Pietro Vertechi. Detailed contributions are provided per chapter in the "Author contributions" section.

Chapter 2 is a published manuscript, in collaboration with Eran Lottem, Dario Sarra, Beatriz Godinho, Isaac Treves, Tiago Quendera, Matthijs Nicolai Oude Lohuis, and Zachary F. Mainen. Chapter 3 is an ongoing work, in collaboration with Mattia G. Bergomi and. Patrizio Frosini. Chapter 4 is a published manuscript, in collaboration with Wieland Brendel, Ralph Bourdoukan, Christian K. Machens, and Sophie Denève.

Artwork courtesy of Shira Lottem and Diogo Matias.

Financial Support

This work was carried out under the International Neuroscience Doctoral Programme (INDP), funded by the Portuguese FCT (Fundação para a Ciência e a Tecnologia, FCT Bolsa PD / BD / 105944 / 2014) and the Fundação Champalimaud.

Contents

Chapter 1

Introduction

Different branches of research have studied intelligent behavior in biological and artificial agents. Determining the adequate course of action in light of partial and uncertain evidence is a challenge that both animals and machine learning frameworks need to overcome. Many approaches exist, and each of them presents its challenges. Here I will discuss how behavioral neuroscience, artificial intelligence, and theoretical neuroscience have tackled this problem.

1.1 Decision-making under uncertainty

The study of human and animal behavior has made great progress since the advent of Behaviorism [120] in the first half of the 20th century. Simple and elegant models of stimulus-response associations [104] were later generalized to sophisticated Reinforcement Learning (RL) algorithms capable of learning almost arbitrary tasks from a scalar reinforcement function [130].

The RL formulation is deceptively simple. An agent explores a finite set of states and can perform one of a finite number of actions. Each action may entail a reward or punishment and leads the agent to a potentially different state. Experiencing the task many times, the agent optimizes

its policy—the rule that associates each state to the corresponding action. Stimulus-response tasks can then be easily interpreted in this framework, where stimuli correspond to states and responses to actions.

However, extending the RL framework from stimulus-response tasks to more general scenarios rapidly encounters a fundamental problem: the dimensionality of the state space is often too large for most RL algorithms. Even when the environment is simple (e.g., a lab environment with only a few stimuli), this dimensionality explosion may happen due to the stimulus dynamics. As the agent's policy is only a function of the state, the state must be maximally informative about future events. Hence, it should encompass all the information concerning the stimulus history that could be predictive of future outcomes. That is problematic when the stimuli are not drawn independently and have a hidden temporal structure. If the last ten stimuli are more predictive of the future than just the last one, then the agent should remember all of them, that is to say, each state should encompass a long stimulus history. As this is often unfeasible, the optimal agent has to discover what features of stimulus history are predictive of the future and encode those in the state representation.

While this is a complex problem in general, it can often be solved in practice using a simple idea. The stimuli the agent experiences may be noisy observations of an underlying Markov process, which is hidden. The state of the RL problem is then the inferred probability of being in each possible state of the Markov process, given the history of observations.

There is evidence that animals can use hidden states to guide behavior, both from the analysis of behavior (see [92] for a review) and neural signals. In particular, the activity of dopaminergic neurons, thought to encode the difference between observed and expected value [113], could actually encode a posterior distribution of possible hidden states in mice [127]. [108] show that primates can learn to infer a contingency change via counterfactual reasoning, and the contingencies themselves are encoded both in

prefrontal cortical regions and in the Amygdala. However, it remains unclear whether such neural signatures are causally linked to the behavioral signatures of inference.

1.2 Intelligible artificial intelligence

Decoding hidden variables is not only a problem for biological agents: it has also been studied extensively in the field of artificial intelligence. The first studies on how to solve decision-making problems under uncertainty originated in the domain of control theory, in the case where not all control variables could be measured [8, 122]. Starting with the work of Cassandra, Kaelbling, and Littman [29, 80], the AI community also became interested in the topic, and efficient algorithms were found in the context of Partially Observable Markov Decision Processes (POMDP) [124]. Even though the POMDP framework is quite abstract, several studies investigated links between inference in POMDPs and motor control [133] or planning [9].

With the advent of Deep Learning (DL), deep neural networks became the preferred method to tackle problems traditionally in the domain of POMDPs [53, 88]. However, while artificial neural networks have been extremely successful in practice, they are not fully satisfactory from a neuroscientific perspective. Even though such over-parameterized networks can generally approximate good solutions to difficult problems, they are often not *intelligible* [141]: designing and training a large deep network on a big dataset (or on simulated data) is a brute-force approach that rarely brings any insight into the nature of the problem. It is hard to deduce, from the optimal parameter values of the fully trained network, which features of the data were used to solve the task, and how they were combined to do so.

Our lack of understanding of the functioning of deep networks has an additional drawback. The choice of architecture has a decisive impact on

the performance of a neural network. Yet there is almost no theory on architecture design that would yield provable guarantees of convergence to the desired solution.

1.3 Biologically plausible learning

It is tempting to think that artificial neural networks (ANNs) are, somehow, a computational model of biological ones. Indeed, convolutional neural networks share many features with the mammalian cortex: receptive fields of simple cells in the primary visual cortex (V1) are qualitatively similar to optimal filters in an efficient coding framework [60, 95]. Of course, on closer inspection, it becomes clear that ANNs lack many of the complexities of biological networks. Most ANNs only consider one cell type, and the dynamics of artificial neurons have often little in common with the biophysical properties of the different cell types of the cortex.

While these problems could probably be fixed by designing more complex ANNs where each neuron has richer dynamics, there remains a striking conceptual issue that concerns learning. A given ANN learns how to solve a task by taking the gradient of an appropriate objective function with respect to its parameters (loosely speaking, the network's synaptic weights). One could be tempted to assume that biological networks might learn in a similar way and that the ANN's learning rule corresponds to some kind of synaptic plasticity. Unfortunately, the quantity that should determine whether each synapse potentiates or depotentiates is often *global* and depends on the activity of all other cells in the network, as well as on the value of the objective function. Do mathematically derived learning rules admit biologically plausible approximations, where the locality constraints of available information are taken into account?

1.4 Overview

The hidden state foraging task

The aim of chapter 2 is to establish that latent variable decoding (inferring hidden causes underlying visible outcomes) can be tested in a laboratory environment, both in humans and in mice, and causally linked to the activity of prefrontal cortical regions. To do so, we developed a task where rewards are probabilistic and depend on a hidden state that changes stochastically. The main challenge, in terms of task design, consists in introducing hidden Markov models while maintaining the environment sufficiently simple.

Structure. Section 2.2.1 describes the *hidden-state foraging task*, a behavioral task that builds on the probabilistic task developed in [81], with two important modifications. On the one hand, it simplifies the structure of the task, so that a single hidden variable (which one of two water ports is rewarding) is a sufficient statistic to describe the task state. On the other hand, the hidden-state foraging task, by design, has a steeper loss function, in that small deviations from the optimal behavior can incur a high cost. As a consequence, the mice's behavior closely matched predictions of the optimal model, which we verified in two ways. First, in section 2.2.2, we show how the inferred probability of the value of the hidden variable governing the task is more predictive of the mice's behavior than simpler quantities such as the reward count. Second, in section 2.2.3, we show that the mice learn to adjust their behavior as a function of the probabilities governing the hidden Markov process. Mice were faster in detecting transitions in protocols where observed events were more informative of the hidden state. In section 2.2.4, we test an adaptation of the task (in the form of a video game) on human subjects and demonstrate that, like their rodent counterpart, humans also match predictions of the normative

model and accumulate evidence rather than rewards. Finally, section 2.2.5, explicitly tests a prediction put forward in [146], where the authors suggest that the ability to base the state representation on quantities that are not directly observable, relies on the Orbito-Frontal Cortex (OFC).

Parametric machines

Chapter 3 serves as a prototype of *intelligible artificial intelligence*. There, we introduce the notion of *machine*—the atomic component of our framework. The definition of machine allows us to link mathematical results on function spaces with fundamental properties of neural networks: modularity and alternation of linear and nonlinear functions.

To construct interesting examples of machines, we adopt two different strategies, corresponding roughly to finite- and infinite-depth architectures. We show how our finite-depth architectures generalize classical neural networks, whereas our infinite-depth architectures generalize neural ordinary differential equations.

To achieve guarantees of optimality, we build on the theory of *operator-valued kernels*. Given operator-valued kernels compatible with finite or continuous filtrations of a Hilbert space X, we define a function space H of continuous endomorphisms of X, such that each $f \in H$ is a machine on X. We can then borrow ideas from the field of kernel methods to select, among all the available small machines in f, the one that is most suitable to solve the computational problem at hand. In our simulations, this approach performs well on small datasets.

Structure. First, we set our categorical foundations. We only need two basic notions to build upon—*machine* and *independence*—which we define in section 3.2.1. Section 3.2.2 discusses finite- and infinite-depth architectures, generalizing classical neural networks and neural ordinary differential equations. In section 3.2.3, we combine kernel methods with the

framework established in sections 3.2.1 and 3.2.2. We formalize the notion of *kernel machine* and give two distinct classes of examples (based on finite and continuous filtrations of Hilbert spaces), which we benchmark on nonlinear regression and classification problems on small datasets.

Learning to represent signals spike by spike

In chapter 4, building on [34], we show that, by minimizing postsynaptic voltage fluctuations, biologically plausible spiking neural networks can learn to encode arbitrary signals robustly and efficiently. In particular, we replace the efficient coding loss function with approximations that aim to either minimize the variance of the postsynaptic membrane potential or to balance the excitatory current received by the postsynaptic neuron.

While the computational problem described in chapter 4 may appear quite simple—as the network only needs to output a signal that it received—sparsity costs make the computation more interesting. In particular, the network can perform latent variable decoding: PCA and ICA can be interpreted as an efficient coding problem with sparsity constrains, as detailed in [76].

Structure. In section 4.2.1, we show that a well-known feature of cortical networks—the balance of excitatory and inhibitory currents [148]—is closely linked with the efficient coding objective. In section 4.2.2, we put forward a possible learning rule for recurrent synapses that would, over time, bring a recurrent network to a tightly balanced stable state, whereas in section 4.2.3, we show that, even after having learned the excitation/inhibition balance, the coding performance can be further enhanced by tuning the feedforward connections to the statistical properties of the input signal. The following sections discuss how to adjust the framework to be more realistic. Section 4.2.4 details how the neural network architecture and learning rules can be adjusted to include separate populations of

excitatory and inhibitory cells. In sections 4.2.5 and 4.2.6, we show that the network, and associated learning rule, do not only work in idealized scenarios, but are robust both to input correlation, random currents, and connectivity constraints. Finally, in section 4.2.7, we discuss experimentally testable predictions of our learning rules.

Chapter 2

Inference based decisions in a hidden state foraging task

Summary

Essential features of the world are often hidden and must be inferred by constructing internal models based on indirect evidence. Here, to study the mechanisms of inference we established a foraging task that is naturalistic and easily learned, yet can distinguish inference from simpler strategies such as the direct integration of sensory data. We show that both mice and humans learn a strategy consistent with optimal inference of a hidden state. However, humans acquire this strategy more than an order of magnitude faster than mice. Using optogenetics in mice we show that orbitofrontal and anterior cingulate cortex inactivation impact task performance, but only orbitofrontal inactivation reverts mice from an inference-based to a stimulus-bound decision strategy. These results establish a cross-species paradigm for studying the problem of inference-based decision-making and begin to dissect the network of brain regions crucial for its performance.

2.1 Introduction

In natural foraging behaviors, animals must continually choose between trying to exploit resources at their current location and leaving to explore another, potentially superior one, at the expense of a possibly costly travel period. Viewed from the perspective of optimal decision-making, the crucial question is when is it best to leave the current site for another one? According to the marginal value theorem, in order to maximize returns, an optimal forager ought to leave its current site when the immediate rate of reward drops below the average rate [31]. However, this elegant solution to the foraging problem only applies in deterministic environments [70], in which both immediate and average reward rates are knowable to the agent. In a more realistic scenario—for example, where rewards are encountered probabilistically—the immediate reward rate is ill-defined and the marginal value theorem does not apply.

One widely-used and powerful approach to model decision-making in dynamic and stochastic environments is reinforcement learning (RL) [130]. In RL, the values of different actions (such as leaving a foraging site or staying on) are continuously updated through trial and error, based on their outcomes, allowing agents to adaptively modify their preferences as conditions change. In its simplest form, model-free RL assigns each action with a value that is updated based on its immediate outcome, with no regard to the causal, and often hidden, structure that links actions to outcomes. While computationally efficient and consistent with a large body of experimental data on both Pavlovian and operant tasks [44, 114], model-free RL is not the best available strategy in many situations. Consider, for instance, a lion that has just successfully captured prey. If the fact that in doing this it has most likely scared away all other animals is ignored, the lion may continue to hunt in the same region, wasting a considerable amount of time searching for the now long-gone prey. Conversely, things

may turn out badly for a zebra if it assumes that its current foraging ground was safe (that is, lion-free) just because it had not seen a lion yet in its immediate surroundings. What these examples illustrate is that relying solely on recent outcomes, while ignoring causal structures in the world, may have suboptimal (if not catastrophic) consequences. Instead, in structure learning [22, 23, 97], a form of inference-based RL, agents choose actions based on their beliefs about the current state of the world, which is determined by both incoming sensory evidence (such as outcomes) and knowledge of the underlying causal structure of the environment. How humans and animals implement such strategies remains an important and poorly understood question [35, 93, 127].

The study of the neural mechanisms underlying flexible, integrative behavior has drawn special attention to the prefrontal cortex and the computational role of its different areas. Although the mapping of the rodent prefrontal cortex has not reached a consensus, here we adopt the description of [137], which defines as rat prefrontal cortex those areas comparable to the primate prefrontal cortex in terms of thalamic reciprocal connections, corticocortical connections, and functional aspects, including the orbitofrontal cortex (OFC) and the anterior cingulate cortex (ACC). It has been suggested that the OFC is crucial for hidden state representation, and hence for inference-based decisions. For example, in both rats and primates, lesions or inhibition of OFC impairs subjects' ability to adjust their behavior in reversal learning tasks, where the depletion of a previously rewarding site (or the futility of a previously rewarding action) may be viewed as a change in the (hidden) state of the world [146], even though this result may be technique-dependent: [106] found that aspiration lesions (which also damage passing fibers), but not excitotoxic lesions, of monkey OFC impaired reversal learning. The adjacent ACC, often considered part of the rodent medial prefrontal cortex (see for example [127, 131]), has been implicated in monitoring value during foraging [54, 69] and could be

responsible for encoding the value of alternative options [71] and changing behavior based on the decreasing value of the current option [118, 145].

Here, we describe a foraging task in which subjects may seek rewards at either one of two foraging sites. This task has a special hidden structure: at any given moment, only one of the sites can deliver rewards and the site of the rewards switches with a certain probability after each foraging attempt. Importantly, even when reward is available, it is not delivered for every attempt, but rather with a probability less than 1. This makes the task a partially observable Markov decision process (POMDP): the true state of the world (i.e., the identity of the rewarding site) is hidden and subjects must infer it based on noisy observations. A defining feature of this task, due to the hidden structure, is the asymmetry of the evidence provided by rewards and failures (unrewarded attempts): a single failure provides partial evidence in favor of a site switch, whereas a single reward provides full certainty that the current site is rewarding. A "stimulus-bound" agent, in the sense of [146], would assign value to observable states (being on the left or on the right or the other foraging site) by linearly combining rewards and failures. Such a process does not capture the essential asymmetry of the task. Ten rewards are much better than one reward in terms of value, but under optimal inference, one single reward is as informative as ten, since it already gives absolute certainty that the current site is active. Thus, leaving decisions under stimulus-bound and inference-based strategies in this task will be qualitatively different. A stimulus-bound agent will become more persistent the more rewards it has received at a site, whereas an inference-based agent will not show such an effect. We found that both mice and humans display hallmarks of inference in the performance of a foraging task and are able to build a non-trivial representation of task space. We further show that optogenetic inhibition of the OFC in mice selectively disrupts optimal inference behavior, biasing mice towards a sub-optimal stimulus-bound strategy. Similar inhibition of

the adjacent ACC, results in delayed leaving decisions but does not disrupt the inference process itself, suggesting a specific role of OFC in this important cognitive function.

2.2 Results

2.2.1 A probabilistic foraging task

We developed a self-paced probabilistic foraging task. Subjects sought rewards by actively probing a foraging site. Each try at the active site yielded reward with probability p_{RWD}, and could cause a switch with probability p_{SW} (fig. 2.1a). After a state switch, to obtain more rewards, subjects needed to travel to a second site at some distance and therefore bear a travel cost. Subjects were thus tasked with inferring a hidden state of the current site through a sequence of observations of stochastic events (rewards and failures). There are actually many ways of integrating rewards and failures to form a decision. In a stimulus-bound process, the relative value of the left site with respect to the right site $V = V_{LEFT} - V_{RIGHT}$ would increase gradually with left rewards, decrease gradually with right rewards, and decay to 0 with failures (fig. 2.1b). In formulas, given a decay coefficient γ, a reward indicator r_t, a site indicator s_t (1 for left, -1 for right), and signed outcomes $o_t = r_t \cdot s_t$:

$$V_{t+1} = (1 - \gamma)V_t + \gamma o_{t+1}. \tag{2.1}$$

On the other hand, an agent that is aware of the structure of the task—the fact that a hidden state determines which site is rewarding at any time—could use rewards and failures differently, allowing it to better infer whether the current foraging site is active or inactive. The relative

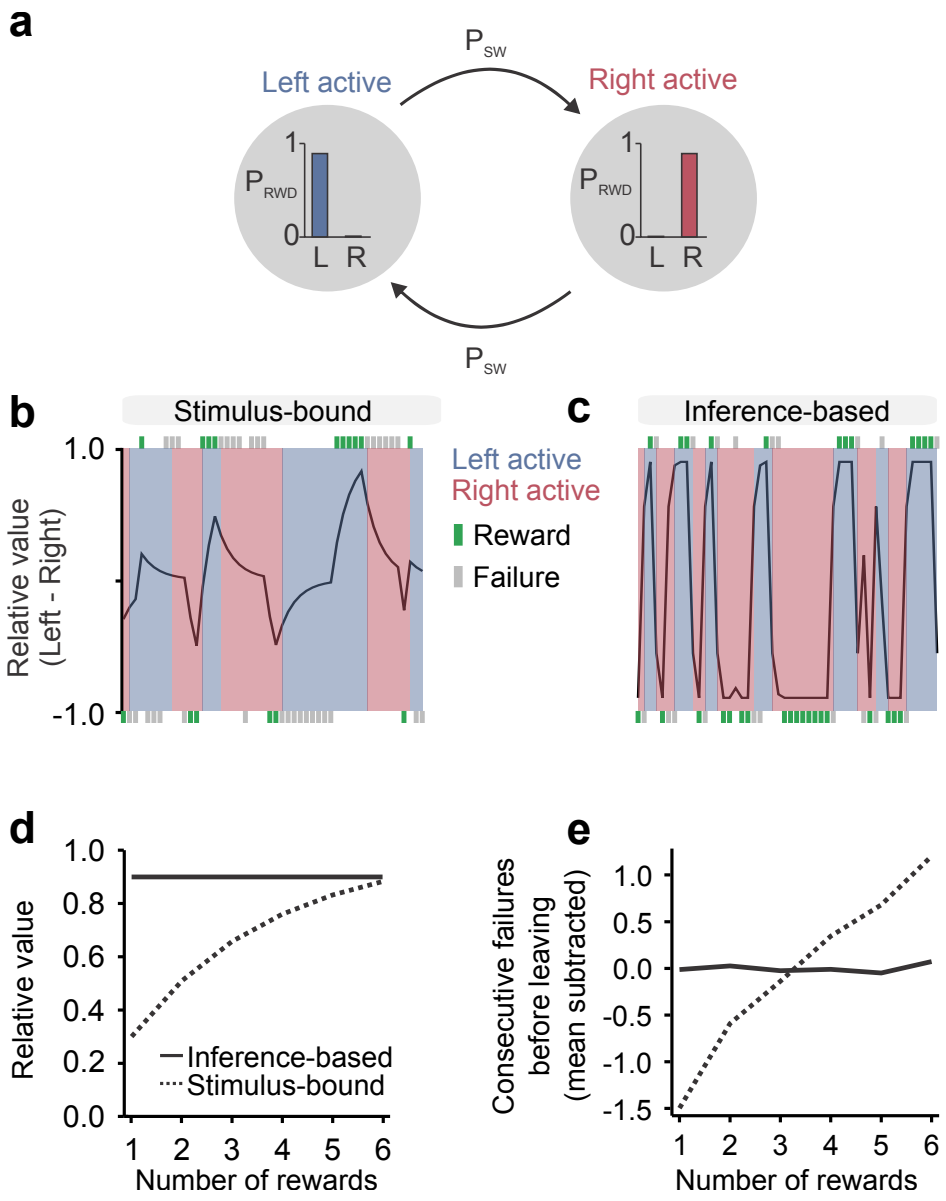


Figure 2.1: (legend on next page)

Figure 2.1: **A Probabilistic Foraging Task Can Dissociate Stimulus-Bound from Inference-Based Evidence Accumulation.** (a) Formally, the task is a hidden Markov model with LeftActive and RightActive states. It has two parameters: probability of reward given state and probability of state transition. (b and c) Estimated relative value (left minus right) as a function of trial history (rewards in green, failures in gray) in the stimulus-bound model and inference-based model respectively. Shaded patches indicate actual state. (d) Effect of rewards on relative value in stimulus- bound and inference-based models: the two models are simulated in a trial with only rewards on the same site. Relative value increases with reward number in the stimulus-bound but not in the inference-based model. (e) Consecutive failures before leaving (normalized subtractively) as a function of reward number in a simulated data of stimulus-bound and inference-based models: reward number has an effect on consecutive failures in the stimulus-bound but not in the inference-based model.

value would then be:

$$V_t = p_{RWD}(P(LeftActive \mid r_1, s_1, \ldots, r_t, s_t) - P(RightActive \mid r_1, s_1, \ldots, r_t, s_t)),$$

(see section 2.4.5 for a detailed treatment of the probability computation). Unlike the stimulus-bound mechanism in fig. 2.1b, this process is able to track effectively the rapidly evolving value of the foraging sites (fig. 2.1c). Both accumulation processes can be used as generative models of the behavior by defining the probability of staying on, e.g., the left site as a sigmoidal function of the relative value:

$$P(NextLeft \mid V, s) = \sigma(\beta(V + s \cdot T)),$$

where β represents a softmax parameter (the higher it is, the more deterministic the behavior), and T represents the staying bias (when the value of the left and the right site are estimated as equal, the subject should still prefer to stay to avoid the travel cost).

Both models predict that the probability of leaving increases with the number of consecutive failures. However, the effect of a reward is very different between them. In a stimulus-bound model the probability of leaving decreases with the number of rewards, as each reward contributes to the accumulated value. In the inference-based model it does not, as a single reward is sufficient to deduce with certainty that the current site is active (fig. 2.1d). Thus a simple test of whether subjects are using inference is to check whether the number of failures before leaving changes with the number of preceding rewards (fig. 2.1e).

2.2.2 Mice accumulate evidence and not rewards

We first developed the hidden state foraging task as a rodent behavioral task (fig. 2.2a) in which mice had to nose-poke at one of two possible ports to obtain water rewards (2 µL each). We trained 18 C57BL/6 wild type mice of 2 months age for 12 days in a baseline protocol with $p_{RWD} = 0.9$ and $p_{SW} = 0.3$ and observed the effect of rewards on behavior during learning.

Since our task was a foraging style task, the mapping to choice and feedback are not present in a trial. Each trial contains a number of pokes, each of which contains feedback in the form of water (or not), as illustrated in fig. 2.2b. The choice of the mouse is whether to continue to poke (exploit) at the current site or whether to leave (explore), a choice it makes after each poke. The only incorrect choice is to leave to the other site before the state has switched. When the mouse switches site too early, no rewards will be emitted by the other port; the mouse is obliged to return to the original port and continue to poke. Mice made only about $2.25 \pm 0.47\%$ ($N = 18$ mice) errors on average. An example of the behavior of a trained animal is shown in fig. 2.2c (see Fig. S1 for summary statistics of the durations of the various task epochs).

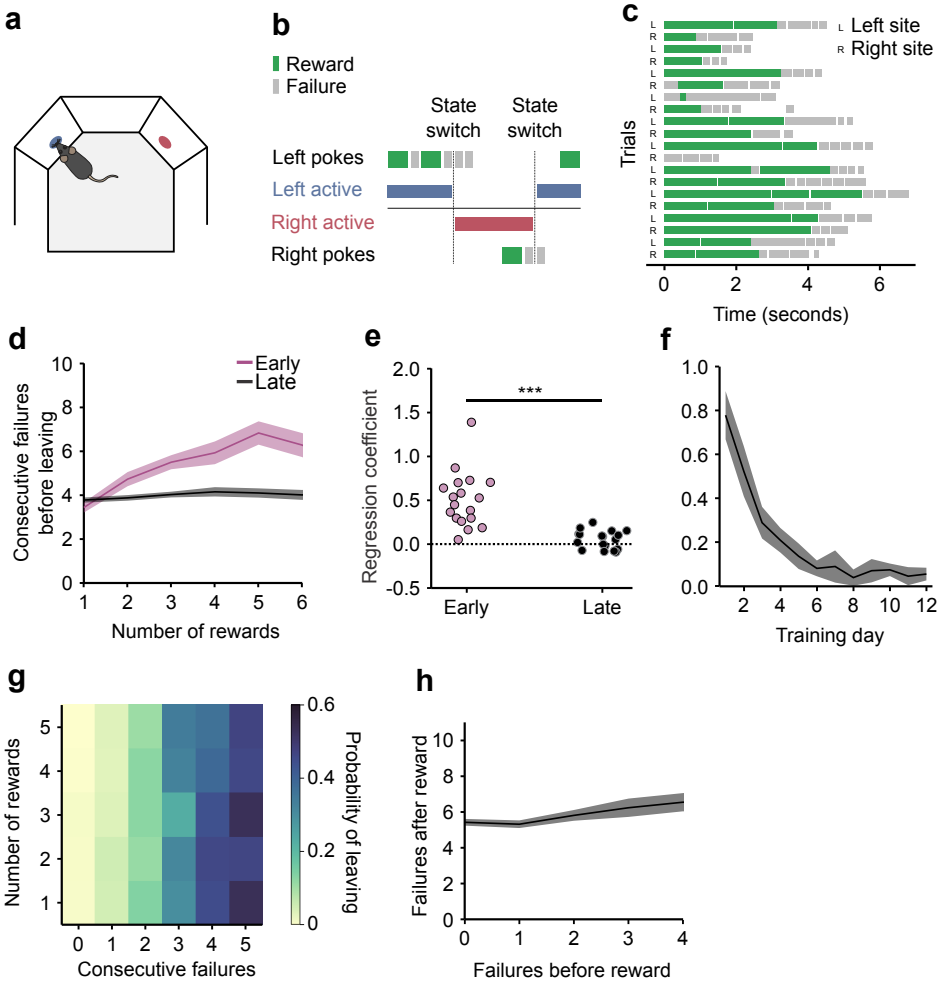


Figure 2.2: (legend on next page)

Figure 2.2: **Mice Accumulate Inferred Evidence for State Switches and Not Site Value.** (a) Schematic of rodent task. Mice shuttle back and forth between two reward sites to obtain water rewards. (b) Example sequence of pokes. Pokes in the correct site can be rewarded or not, whereas pokes in the incorrect site are never rewarded. Following a state switch, the animals need to travel to the other site to obtain more rewards. (c) Example behavior: sequence of poke bouts (i.e., trials) with rewards in green and failures in gray. (d) Consecutive failures before leaving as a function of reward number in early training (days 1 to 3, purple) compared with late training (days 10 to 12, black). (e) Slope coefficient in $ConsecutiveFailures \sim 1 + RewardNumber$ for early training and late training. Slope coefficient is higher in early trials, likelihood ratio test on linear mixed-effect model $ConsecutiveFailures\ 1 + RewardNumber + Early + RewardNumber\&Early + (1 \mid MouseID)$ versus a null model with no interaction: $p < 10^{-10}$, $N = 18$ mice (see section 2.4.5 for a description of the formula notation). (f) Evolution of reward number coefficient across days. (g) Probability of leaving as a function of number of rewards and consecutive failures in late training. (h) Failures after reward as a function of failure before reward in trials with only one reward in a more difficult protocol.

Mice tended to alternate bouts of pokes at a given site (6.98 ± 0.14 pokes per bout, $N = 18$ mice) with trips to the opposite site, producing a natural segmentation in trials (i.e. poke bouts on the same site). This presumably reflects the clear asymmetry in time cost between nose-poking again on the same site, a very cheap action (Fig. S1, inter-poke interval = 0.16 ± 0.025 seconds, unrewarded poke duration = 0.33 ± 0.006 seconds, $N = 18$ mice), and switching site, a much more expensive option (Fig. S1, 3.15 ± 0.18 seconds, $N = 18$ mice).

We found that the number of consecutive failures since the last reward (ConsecutiveFailureIndex) was a better predictor of mouse choice than the time spent at the nose poke (TimeSpentAtPort) (Lottem et al., 2018). We fitted two logistic regression models with random effects. Here and throughout the text we use Wilkinson notation [143] (see section 2.4.5 for

a detailed explanation):

$$LeavingPort \sim 1 + ConsecutiveFailureIndex + (1 \mid MouseID)$$

and

$$LeavingPort \sim 1 + TimeSpentAtPort + (1 \mid MouseID).$$

The ConsecutiveFailureIndex model was overwhelmingly better (deviance = 10892) than the TimeSpentAtPort model (deviance = 14771). As confirmation, we also tested a model that included both predictors:

$$\begin{aligned} LeavingPort \sim & 1 + ConsecutiveFailureIndex + TimeSpentAtPort \\ & + (1 \mid MouseID), \end{aligned}$$

and only the ConsecutiveFailureIndex had a positive coefficient (0.78 ± 0.013) whereas the TimeSpentAtPort had a small negative coefficient (-0.046 ± 0.006).

In the early part of training, animals were unaware of the structure of the task and exhibited hallmarks of a stimulus-bound strategy: more failures were needed to leave the foraging port in rich foraging bouts, with many rewards before a state switch, compared to poor foraging bouts, with as little as one reward before a state switch. After training, however, the number of rewards had no effect on the number of failures before leaving, consistent with an inference-based strategy (fig. 2.2d). To quantify this effect at a single animal level, we fitted a linear regression model that predicted the number of consecutive failures before leaving as a function of the number of prior rewards in the current trial (i.e. foraging bout at a given site): $ConsecutiveFailures \sim 1 + RewardNumber$ (fig. 2.2e). The data show that during the first days of training there was a strong positive correlation between these two quantities, but with continued training this

correlation decayed to zero (fig. 2.2f). Therefore, experienced mice, unlike naïve animals, decide when to leave the foraging site in a manner consistent with inferring a hidden state, rather than directly integrating rewards and failures.

As another way of seeing this, a stimulus-bound integration strategy would effectively weigh similarly each reward and failure with opposite signs (see eq. (2.1)). Correct inference instead, given the structure of this task, requires that rewards are weighted nonlinearly (the first counting a lot and subsequent nothing) and differently from failures, which should add linearly. Indeed, in the trained mice, the effect of rewards and failures in shaping the behavior is qualitatively asymmetric in just this way, as can be seen by visualizing the probability of leaving as a function of both reward number and consecutive failures (fig. 2.2g).

Furthermore, stimulus-bound and inference-based models predict different interactions of rewards with preceding failures. Consider, for example, trials in which the animal receives a single reward: the later the reward, the smaller the value of the current site at the time of reward delivery. In the stimulus-bound model, the received reward value is simply added to the current value estimate, so the later in the train the reward arrives the lower the current value of the port (given that we are assuming only failures before this reward). Therefore, fewer subsequent failures will be tolerated before the animal leaves. On the other hand, in inference-based models a single reward resets the count of accumulated failures up to that point, and therefore the position of the reward (or equivalently the number of failures prior to the reward) has no consequence on subsequent behavior. To test these alternatives, we analysed how the position of that reward influenced the overall number of failures before leaving in a protocol with lower probability rewards ($p_{RWD} = 0.3$ and p_{SW}) and found that the number of failures after the last reward did not decrease when it was preceded by more and more failures, on the contrary it slightly increased

(fig. 2.2h, slope $= 0.2 \pm 0.06$, $N = 20$ mice), consistent with a resetting effect of reward as predicted by the inference-based model.

2.2.3 Accumulation of evidence is tuned to task parameters

Having found that the foraging behavior of mice is consistent with the accumulation of evidence to infer a hidden world state, we asked whether this inference process is appropriately tuned to the statistics of the foraging environment, represented here by two parameters: reward probability p_{RWD} and state switch probability p_{SW}. Intuitively, if is high, then a single failure is strong evidence in favor of a state switch, leading to a faster accumulation process. Similarly, if is high, then a failure also carries more evidence in favor of a state switch compared to if it is low (fig. 2.3a; see section 2.4.5 for a formal justification of this intuitive argument).

To test this, we trained a separate batch of mice on a set of three different foraging site statistics (Easy environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.9$; Medium environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.3$; Hard environment: $p_{RWD} = 0.3$ and $p_{SW} = 0.3$; see fig. 2.3b). Since changing the foraging environment's statistics can affect average reward rates (i.e. average number of rewards per trial), we adjusted the magnitude of individual rewards in order to equalize the amount of reward at a given site before state switch across conditions. As predicted normatively, mice increased the number of failed attempts they would tolerate as the state switching probability and the reward probability dropped (fig. 2.3c, d; difference in failed attempts after last reward in Easy-Medium $= -1.61 \pm 0.03$, difference Hard-Medium $= 1.77 \pm 0.04$, $N = 20$ mice, likelihood ratio test on $ConsecutiveFailures \sim 1 + Protocol + (1 \mid MouseID)$ versus $ConsecutiveFailures \sim 1 + (1 \mid MouseID) : p < 10^{-10}$).

An important additional prediction of optimal decision theory in the context of a foraging task is that travel cost should modulate the threshold to leave a given foraging site. To test this, we increased the travel

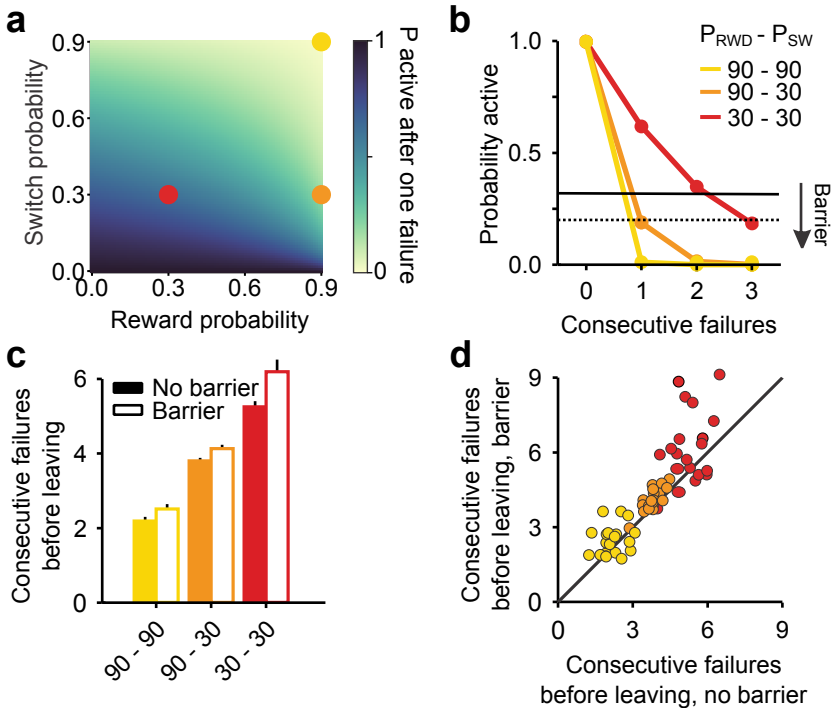


Figure 2.3: **Accumulation of Inferred Evidence Is Tuned to Task Parameters.** (a) Probability of being on the correct site after a failure as a function of reward probability and transition probability. (b) Probability of being on the correct site as a function of trial history for three protocols (Easy environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.9$; Medium environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.3$; Hard environment: $p_{RWD} = 0.3$ and $p_{SW} = 0.3$). Leaving decisions can be modeled by setting a threshold on this probability that changes as a function of the travel cost (black lines). (c) Consecutive failures before leaving as a function of the environment statistics and barrier condition. (d) Consecutive failures before leaving split by subject and environment statistics, barrier versus no barrier.

cost by placing a physical barrier between the two locations (travel time without barrier = 1.86 ± 0.13 s, $N = 20$ mice; travel time with barrier = 2.69 ± 0.13 s). Once again, the accumulation process was modulated consistently with the normative prediction, longer travel times resulting in a longer accumulation process and delayed leaving (fig. 2.3c, d, effect of barrier in number of failed attempts after last reward = 0.42 ± 0.03, $N = 20$ mice, likelihood ratio test on $ConsecutiveFailures \sim 1 + Protocol + Barrier + (1 \mid MouseID)$ versus $ConsecutiveFailures \sim 1 + Protocol + (1 \mid MouseID)$: $p < 10^{-10}$).

2.2.4 Humans perform inference and tune behavior to task parameters

To test whether our findings were valid across species, we developed a translation of our behavioral assay for human subjects, in the form of a video game, where players would drag a character from one side of a touch screen to the other and tap to achieve points. The statistics of the video game (p_{RWD} and p_{SW}) were the same as those used in the rodent task.

In humans we again observed hallmarks of inference-based foraging: the number of rewards had little to no effect on the behavior (Fig 4a, b), similar to the behavior of the trained mice. Unlike mice, however, humans needed almost no training to learn this strategy, displaying it from the first session.

Analogously to their rodent counterparts, human subjects modulated their behavior according to reward statistics as well as travel time (here affected by a manipulation in the character's velocity) consistent with the normative predictions (fig. 2.4c, d, difference in failed attempts after last reward in Easy-Medium = -1.39 ± 0.03, difference Hard-Medium = 1.48 ± 0.03, $N = 20$ subjects, likelihood ratio test on $ConsecutiveFailures \sim 1 + Protocol + (1 \mid SubjectID)$ versus $ConsecutiveFailures \sim 1 + (1 \mid SubjectID)$: $p < 10^{-10}$, effect of

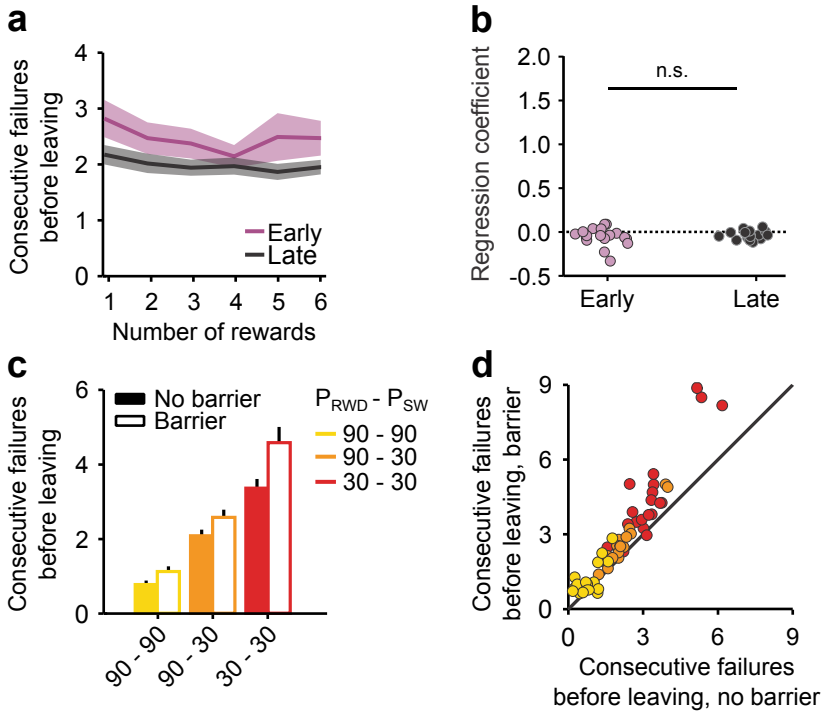


Figure 2.4: **Humans Perform Optimal Inference and Tune Behavior to Task Parameters.** (a) The number of rewards has little effect on the probability of leaving during both early (purple) and late (black) training. (b) Number of consecutive failures as a function of reward number for human in early versus late part of training. Unlike mice, humans learn the statistics of the environment extremely quickly: slope coefficient is similar (and around 0) in both early and late trials: likelihood ratio test on linear mixed-effect model $ConsecutiveFailures \sim 1 + RewardNumber + Early + RewardNumber\&Early + (1 \mid SubjectID)$ versus a null model with no interaction: $p = 0.45$, $N = 20$ subjects. (c) Consecutive failures before leaving as a function of the environment statistics and barrier condition. (d) Consecutive failures before leaving split by subject and environment statistics, barrier versus no barrier.

barrier in number of failed attempts after last reward = 0.59 ± 0.02, $N = 20$ subjects, likelihood ratio test on $ConsecutiveFailures \sim 1 + Protocol + Barrier + (1 \mid SubjectID)$ versus the model with no barrier $ConsecutiveFailures \sim 1 + Protocol + (1 \mid SubjectID)$: $p < 10^{-10}$).

2.2.5 OFC, but not ACC, is necessary for the correct inference process

Finally, to study the brain mechanisms of inference in this task, we tested the involvement of different regions of prefrontal cortex by silencing them using optogenetic stimulation of inhibitory GABAergic interneurons in VGAT-ChR2 mice (mice expressing the excitatory opsin channelrhodopsin-2 in inhibitory GABAergic neurons). We examined 19 mice. Nine were bilaterally implanted with optic fibers (table S1) in the anterior cingulate cortex (ACC; fig. 2.5a, S2a), six of these mice were ChR2-expressing (HET) and three were control wild-type littermates (WT) implanted and stimulated in the same manner. Ten (six HET and four WT) were bilaterally implanted in the orbitofrontal cortex (OFC; fig. 2.5a, S2b). Transient inactivation of ACC (3 mW power per fiber, 10 ms pulses at 75 Hz, during poking; triggered by the first poke in 50% of trials and maintained for 500 ms after each poke in the trial; fig. 2.5b) significantly increased the average number of consecutive failures before leaving (fig. 2.5c effect of stimulation on consecutive failures after last reward = 0.48 ± 0.05, N = 6 mice, likelihood ratio test on versus : p < 1e-10). The same protocol applied to control mice had no effect (fig. 2.5e, effect of stimulation = 0.003 ± 0.07, N = 7 mice, likelihood ratio test: p = 0.96). More specifically, we found that ACC inactivation multiplicatively increased the number of consecutive failures before leaving, consistently across protocols and animals (fig. 2.5f, and interact when predicting , likelihood ratio test: p = 1.46e-6, but not when predicting renormalized , likelihood ratio test: p = 0.15, N = 6 mice). Transient inactivation of OFC also increased the

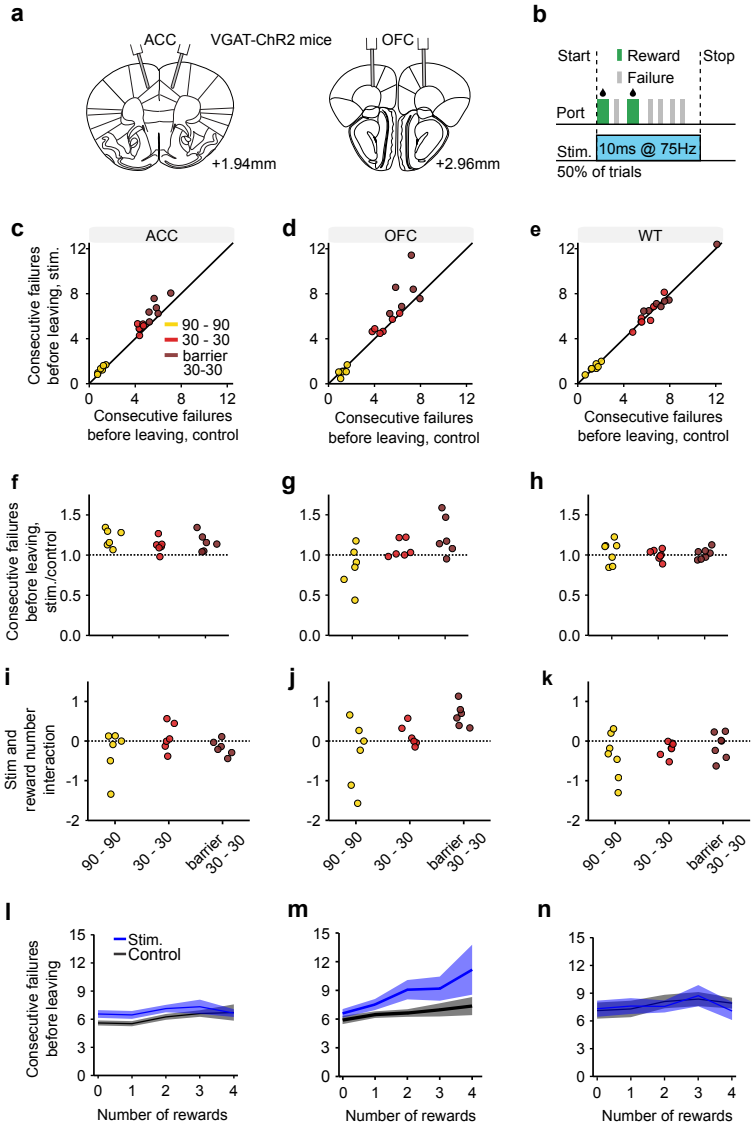


Figure 2.5: (legend on next page)

Figure 2.5: **OFC, but Not ACC, Is Necessary for Optimal Inference.** (a) Scheme of the optic fiber placement. (b) Bilateral photostimulation at 3 mW happened during nose-poking: it was triggered by the first poke in 50% of trials and lasted for 500 µs after the last poke in. (c–e) Consecutive failures before leaving split by environment statistics, barrier condition, and subject inactivation versus control trials, for ACC implanted heterozygotes (c), OFC implanted heterozygotes (d), and wild-types (e), respectively. (f–h) Ratio of consecutive failures before leaving split in the same way as (c)–(e) for ACC implanted heterozygotes (f), OFC implanted heterozygotes (g), and wild-types (h), respectively. When predicting renormalized consecutive failures, Stimulation and Protocol interact for OFC implanted heterozygotes ($p < 10^{-10}$, $N = 6$ mice) but not for ACC implanted heterozygotes ($p = 0.15$, $N = 6$ mice) or wild-types ($p = 0.77$, $N = 7$ mice). (i–k) An animal by animal quantification: the coefficient of the interaction term in $ConsecutiveFailures \sim 1 + Stimulation + Rewardnumber + Rewardnumber\&Stimulation$ for ACC implanted heterozygotes (i), OFC implanted heterozygotes (j), and wild-types (k). (l–n) Number of consecutive failures as a function of reward number in the 30-30 barrier protocol for ACC implanted heterozygotes (l), OFC implanted heterozygotes (m), and wild-types (n).

average number of consecutive failures before leaving (fig. 2.5d; effect of stimulation on consecutive failures after last reward = 0.41 ± 0.08, N = 6 mice, likelihood ratio test on versus : p = 5.38e-7). However, unlike the case for ACC inactivation, it did so in a manner that was dependent on the statistics of the environment. That is, the direction of effect for OFC inactivation actually reversed between easy and difficult protocols (effect of stimulation in hard protocol = 1.52 ± 0.24, effect of stimulation in easy protocol = -0.19 ± 0.05, N = 6 mice). This suggests that these two brain areas are differentially involved in the task. To further investigate the involvement of these prefrontal areas in the inference process and task space representation, we considered a key difference in prediction between the inference model and the simpler stimulus-bound model: the effect of rewards on behavior. As noted above, under normal conditions, rewards fully reset the accumulation process, so that leaving times are not affected by the

number of previous rewards (fig. 2.1e). Strikingly, we found that OFC, but not ACC inactivation, disrupted this pattern: in OFC-inactivated trials, animals became sensitive to the number of rewards: the more rewards gained, the more delayed leaving decisions became (fig. 2.5i, j; for ACC, interaction effect of stimulation and reward number $= -0.038 \pm 0.07$, N = 6 mice, likelihood ratio test on versus

p = 0.58; for OFC, interaction effect of stimulation and reward number $= 0.36 \pm 0.1$, N = 6 mice, likelihood ratio test: p = 0.0003; triple interaction term of stimulation, reward number and fiber location $= 0.47 \pm 0.11$, likelihood ratio test: p = 3.44e-5). This pattern of behavior (illustrated in fig. 2.5l, m) is similar to the one observed in naïve mice first introduced to this task (fig. 2.2d, e), and is indicative of a less effective stimulus-bound strategy. Thus, the OFC is crucial for behavioral strategies in foraging environments in which states are hidden and require inference based on noisy observations.

2.3 Discussion

In this study, we developed a task in which subjects had to alternate between two foraging sites, only one of which was active at any given moment. The task embodied an important form of non-sensory uncertainty because the active port only delivered rewards with a certain probability. The task thus required subjects to infer whether each omitted reward was simply a stochastic failure or was instead an actual switch of state, offering us a way to directly test whether they have the ability to perform state inference. To solve this task optimally, subjects were essentially required to infer a hidden state of the world (i.e. which site is active) rather than directly assigning a value to each foraging site, as would be optimal, for example, in a matching task [57, 129]. We found that both mice and

humans displayed hallmarks of optimal, inference-based behavior, reaching very similar solutions.

Our analysis of the behavioral data, particularly the number of consecutive non-rewarded tries before leaving, revealed that leaving decisions agreed with normative predictions of an inference-based foraging strategy in four important ways: (1) the number of consecutive failures was positively correlated with the propensity to leave; (2) rewards had a resetting effect on the leaving decision process; (3) subjects were sensitive to quantitative changes in the statistics of the foraging site; and (4) subjects were sensitive to the travel cost. However, mice and humans differed in an important way: while it took around six days for rodents to understand the environment statistics and integrate trial history correctly, humans started displaying hallmarks of the optimal behavior already during the first session. This difference may be due to faster learning but could also reflect the ability to generalize prior structural knowledge relevant to the task.

The accumulation of evidence is considered a primary cognitive computation. Similarly to sensory-guided tasks, in which integration of sensory evidence over time is needed to "average out" stimulus noise [24, 51, 117], here too repeated sampling is needed to determine which of two sites is currently rewarding. Specifically, each failure conveys ambiguous information, as it may be due to either an unlucky attempt at the rewarding site, or a guaranteed failure in the non-active site, and it is only by counting (integrating) the number of consecutive failures that a more accurate state estimation can be made. Our analysis of the leaving probability revealed that, much like in sensory-based tasks, subjects do integrate this information when deciding whether to stay or leave. Moreover, by changing reward and transition probabilities, we were able to precisely control the amount of information associated with each failure and observed that subjects readily adapted their leaving decisions to these changing conditions,

such that the lower the information content of each sample was, the more such samples were needed before leaving.

In the framework of reinforcement learning under uncertainty, given the entire task history—i.e. the sequence of rewards and failures at each port since the beginning of the task—the optimal agent needs to compute a low-dimensional state representation [92] that is most informative of future events. For a formal definition, see [130, Sect. 17.3]. We considered two distinct algorithms for doing this. In the 'inference-based' algorithm, we hypothesize that animals recover a meaningful state representation that allows them to take the most advantage of the task structure. The current state is represented by the posterior probability of being at the active site given the task history, which in practice is a function of the number of consecutive failures since the last reward. In this algorithm, the final learned solution is optimal and independent of the learning rate used during training. Alternatively, in the 'stimulus-bound' algorithm, the animal only uses observable states based on currently available perceptual information [146]. The entire task history is summarized by the current location of the animal (left or right site). In the hidden state task, this representation only allows for suboptimal stable solutions, i.e. policies that depend on the site, but not on the reward history.

A primary distinction between sub-optimal, stimulus-bound and optimal, inference-based strategies lies in the impact consecutive rewards have on leaving decisions. In a stimulus-bound behavior, which assigns values directly to the foraging site, the more consecutive rewards are gained at a given site, the higher the value of staying becomes, and consequently, leaving decisions tend to be delayed. In contrast, optimal inference in this task requires ignoring the number of consecutive rewards, since the delivery of a single reward is sufficient to know for certain which site is currently rewarding. As shown in fig. 2.2, we found that, although initial behavior appeared to be sensory bound, after learning, subjects' leaving decisions became in-

dependent of the number of rewards, consistent with an inference-based approach to leaving decisions. We presume that the change in behavior (between stimulus-bound and inference-based decisions) over the course of training reflects learning, but not that the change in performance necessarily reflects a change in learning rate. What we posit is that two different behavioral controllers, one stimulus-bound and one inference-based, exist, which might correspond to a striatal system and a prefrontal system respectively. Over the course of training, the inference-based controller learns the structure of the task—the correct state representation—through a slow process. In parallel, the inference-based controller's contribution to behavioral choices is increased over training. This scheme is similar to what was proposed by [36]. Alternatively, the change in behavior over time could be accomplished by meta-learning of hyper-parameters: a more complex stimulus-bound agent could keep track of two different learning rates, one for reward and one for failures. With training, the agent would learn that the optimal reward learning rate is one (complete reset) whereas the failure learning rate is adjusted over time to account for different protocols. Even though this algorithm is distinct, it still requires the ability to adjust a failure learning rate in such a way that it is big for informative failures (in easy protocols) and small for uninformative ones (in harder protocols). Consequently, it results in a computation analogous to the inference model.

Recent accounts [92, 112, 126, 146] proposed that the OFC is crucial for accurate state representations, particularly when states are hidden (that is, not explicitly given by the presence of a sensory cue, for example) and have to be inferred from fuzzy evidence. Our findings mesh well with this theory, as we found that OFC, but not ACC inhibition disrupted inference-based behavior. Unlike under control conditions, in which the number of failures before leaving was independent of the number of previously gained rewards, inhibiting the OFC resulted in mice performing more failures when experiencing large amounts of reward. This latter pattern

is consistent with a stimulus-bound strategy and suggests the possibility that the stimulus-bound strategy serves as a default behavioral approach, and is suppressed by the OFC when inference-based behavior is required. The observation that naïve mice behave very similarly to OFC-inactivated mice supports this idea. Several specific computational roles of OFC could account for this effect. OFC could be encoding the representation directly, or be necessary to access or update such representation. Alternatively, the representation could be still available, even when the OFC is inactivated, but the region would be responsible for computing the posterior of the hidden states given the representation.

The ACC has been implicated, in Pavlovian and operant tasks, as a potential candidate for the implementation of integration-to-threshold models. In [66], the authors observe neurons in the primate ACC whose firing rate scales with the number of consecutive negative outcomes in a Pavlovian task. In the setting of an operant task, [109] showed that such negative outcome accumulation is modulated by the error type (the more surprising the error, the stronger the response) and that microstimulation of ACC accelerates the detection of a context switch. From the foraging perspective, [54] reported cells in the ACC encoding the value of a depleting option. However, ACC inactivation in our task, unlike OFC inactivation, had only a modulatory effect on behavior: we did not observe qualitative changes in the strategy of the animals, but only an overall tendency to stay longer at the current port, which interacted multiplicatively both with the task statistics and with increased travel times. The potential activation of neurons in regions immediately adjacent to ACC, e.g. prelimbic cortex or secondary motor cortex, is possible (Fig. S2c). However, the areas targeted in the two experiments (ACC and OFC) are considerably further apart (Fig. S2c). Since we observed a double-dissociation of effects it is unlikely that the fields of neurons activated across these two experiments were substantially overlapping. Although caution is required in comparing

primate and rodent ACC, given the reported anatomical [137] and functional differences [90, 115], our results seem most compatible with the idea that the ACC encodes the value of alternative options, as proposed in [71], while not having a primary role in the computations required for the state inference process.

By developing a human video-game and a rodent task requiring the same underlying computation to be solved, we could compare computational and cognitive processes across species. From a theoretical standpoint, this strengthens the generality of those results that held true for the two species, such as the ability to infer the hidden structure of the environment and to tune behavior to environmental statistics. From a practical standpoint, the hidden state foraging task makes it possible to use rodent experimentation to more closely guide human clinical research into the mechanisms of manipulations (e.g. drugs) or conditions (e.g. depression) that may affect processes such as state inference.

2.4 Materials and methods

Table S1, figures S1,2, and the supplementary videos are available online at the following URL: `https://www.sciencedirect.com/science/article/pii/S089662732030043X#app2`.

2.4.1 Key resources table

REAGENT or RESOURCE	SOURCE	IDENTIFIER
Chemicals, Peptides, and Recombinant Proteins		
DAPI	SIGMA ALDRIC	Cat#D9542; RRID:AB_2801570
Experimental Models: Organisms/Strains		
C57BL/6NCrl	Charles River Laboratories	strain code: 027; RRID:IMSR_CRL:475
Dat-Cre	Jackson Laboratory	stock number: 006660; RRID:IMSR_JAX:006660
Gad-Cre	Jackson Laboratory	stock number: 010802; RRID:IMSR_JAX:010802
VGAT-ChR2	Jackson Laboratory	stock number: 014548; RRID:IMSR_JAX:014548
FI12-Cre	Mutant Mouse Regional Resource Centers	stock number: 017262-UCD; RRID:MMRRC_017262-UCD
Sert-Cre mouse line 61	Mutant Mouse Regional Resource Centers	stock number: 017260-UCD; RRID:MMRRC_017260-UCD
Software and algorithms		
ImageJ	Schneider et al., 2012	`https://imagej.nih.gov/ij/`
Construct 2	Scirra Ltd.	`https://www.scirra.com/construct2`
Julia language	Bezanson et al., 2017	`https://julialang.org/`
MixedModels.jl	Bates et al., 2019	`https://github.com/JuliaStats/MixedModels.jl`
Other		
Arduino Mega 2560 r3	Arduino	A000067
Pokes detector and valve controller	Champalimaud Hardware Platform	Mice poke simple v1.1
Arduino ports interface	Champalimaud Hardware Platform	Arduino baseboard v2.2

2.4.2 Lead contact and materials availability

Further information and requests for reagents may be directed to, and will be fulfilled by the Lead Contact, Zachary F. Mainen. This study did not generate new unique reagents.

2.4.3 Experimental model and subject details

Mice

Fifty-seven adult male C57BL/6 mice were used in this study. For the inference-based versus stimulus-bound behavior experiment (fig. 2.2) 18 C57BL/6NCrl wild-type mice of two months age were used. For the protocols manipulation experiment (fig. 2.3), 20 wild-type animal from different genetic backgrounds (8 Dat-Cre ; 5 Gad2-Cre; 5 Sert-Cre; 1 VGAT-ChR2; 1 F512-Cre) of 6-8 months age were used, in order to reduce animal usage. For inactivation of anterior cingulate or orbitofrontal cortices (fig. 2.5), 12 VGAT- ChR2 and 7 wild-type littermates were used. Mice genotypes were determined based on PCR and further verified using histological inspection of YFP expression which led to the exclusion of a single ACC im-

planted animal from further analysis (see fig. S2d, e). The C57BL/6NCrl line was obtained from the Charles river laboratories, breeders were ordered and bred in-house for a maximum of 4 generations or 2 years (strain code: 027). The Dat-Cre mouse line was obtained from the Jackson laboratory (stock number: 006660). The Gad2-Cre was obtained from the Jackson laboratory (stock number: 010802). The Sert-Cre mouse line 61 was obtained from the Mutant Mouse Regional Resource Centers (stock number: 017260-UCD). The VGAT-ChR2 mouse line 8 was obtained from the Jackson laboratory (stock number: 014548). The FI12-Cre mouse line was obtained from the Mutant Mouse Regional Resource Centers (stock number: 017262-UCD).

All experimental procedures were approved and performed in accordance with the Champalimaud Centre for the Unknown Ethics Committee guidelines and by the Portuguese Veterinary General Board (Direção-Geral de Veterinaria, approval 0421 / 000 / 000 / 2016). The mice were kept under a normal 12 h light/dark cycle, and training, as well as testing, occurred during the light period. Before testing or after surgeries, for the inactivation experiments, mice were single-housed. During training and testing the mice were water deprived, and water was available to them only during task performance. Food was freely accessible to the mice in their home cages. Extra water was provided if needed to ensure that mice maintain no less than 80% of their original weight. For the protocols manipulation experiment behavioral training lasted 12 sessions, once per day, followed by 2 days of rest at the end of which we commenced testing. During training mice were exposed to the 3 different protocols (Easy environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.9$; Medium environment: $p_{RWD} = 0.9$ and $p_{SW} = 0.3$; Hard environment: $p_{RWD} = 0.3$ and $p_{SW} = 0.3$) for 4 consecutive days (1 day of adaptation and 3 of testing) before transitioning to the next environment. During testing, mice performed 1 session per day, 6 or 7 days a week. In protocols manipulation experiments, the sequence

of protocols was counterbalanced across 2 groups of 10 mice (Group A: Hard, Medium, Easy; Group B: Easy, Medium, Hard). In our analyses we considered 50 poke bouts per session after the first 10 during testing days and excluded poke bouts with no rewards.

Human participants

20 right handed healthy adults of Portuguese nationality (10 female and 10 male; 22 to 31 years of age), with no history of psychiatric diagnosis or prescribed drugs in the last 6 months, participated in this study. All participants gave written informed consent, and the study was conducted in accordance with the guidelines of the local ethics committee. The task consisted of 2 sessions of 1 hour, performed in different days with 2 to 10 days in between sessions. Each session consisted of 4 blocks with different protocols, and 10 minutes break after the second block. The sequence of protocols consisted of a block (Medium environment) followed by a short break (2 minutes), then a second block (Easy or Hard environment) followed by a long break (10 minutes), then a third block (Medium environment) followed by a short break (2 minutes) and a final block (Hard or Easy environment). The sequence of environments during testing was counterbalanced across 2 groups as described in mice experiments. In our analyses we considered all tapping bouts after the first 10 and excluded bouts with no rewards.

2.4.4 Method details

Mice behavioral apparatus

The behavioral apparatus for the task was adapted from the design developed by Zachary F. Mainen and Matt Recchia (Island motion corporation, Nesconset, NY), originally developed for rat behavior. The behavioral box ($15 \times 12 \times 18\,\text{cm}$, model 003102.0001, Island motion corporation),

contained 3 front walls (135-degree angle between the center and the side walls) with 2 nose-poke ports attached to the left and right front walls. For the inference-based versus stimulus-bound behavior experiment (fig. 2.2), we used a custom-made acrylic replicate of the box ($15 \times 16 \times 20\,\text{cm}$).

Each port was equipped with infrared emitter/sensor pairs to report the times of port entry and exit (model 007120.0002, Island motion corporation). A nose-poke was considered valid if the infrared beam was broken for at least 100 ms. Water valves (LHDA1233115H, The Lee Company, Westbrook, CT) were calibrated to deliver a drop of 6 µL water for rewarded pokes in Easy and Hard environments and 2 µL of water in Medium environment: the reward size was adjusted to keep the reward amount per correct trial constant. The average number of rewarded attempts per correct trial is $\frac{p_{RWD}}{p_{SW}}$, that is to say 1 in the easy and hard protocol (reward magnitude = 6 µL, amount of water per correct trial = 6 µL) and 3 in the medium protocol (reward magnitude = 2 µL, amount of water per correct trial = 6 µL). In optogenetic experiments, all protocols had an average of one reward per trial, but the reward size was kept at 4 µL to increase the trial number. In optogenetic experiments, blue LEDs were placed in the box ceiling and in all the ports to deliver a masking light.

All signals from sensors were processed by the Arduino Mega 2560 microcontroller board (Arduino, Somerville, US) and output from the Arduino Mega 2560 microcontroller board was implemented to control water and light delivery. Arduino Mega 2560 microcontroller was connected to the sensors and controllers through an Arduino Mega 2560 adaptor board developed by the Champalimaud Foundation Scientific Hardware Platform. An example behavioral video is available in the supplemental information.

Human video game task

Human subjects played a video game on a touchscreen device, with analogous features to the rodent behavioral assay. In the game, subjects receive verbal instructions on how to control a character—a "witch"—on its quest to find and defeat an enemy that hides behind a castle. The witch must walk along the wall of a castle, shooting either the left or the right edge of this wall in search of the enemy that hides, at any given moment, in one of these two edges. The game obeys the same statistics as the rodent task: hitting the enemy is analogous to a water reward, the current location of the enemy corresponds to the active site, and every shot at the active site hits the enemy with probability p_{RWD}. Moving between the two sides of the wall has an associated cost (travel cost) that can also be manipulated with the appearance of rougher terrain (analogous to the physical barrier) that diminishes the traveling speed. As in the mouse case, reward size was manipulated to keep the average reward per correct bout constant (3 points). The game ended either when subjects collected 280 points or when a time limit of 20 minutes was exceeded. Different environments had minor changes in the background images between them—for the medium protocol since it was experienced twice per session, two different backgrounds were used. After the player transitioned to a different site, the enemy was displayed to cue whether the transition had been correct, or if instead the player had to return to the previous site.

The human task was made using custom software developed using the game engine Construct2 (Scirra Ltd., Studio 117, The Light Bulb 1 Filament Walk Wandsworth, London, UK). Graphics were made by Shira Lottem and Tiago Quendera using Inkscape: Open Source Scalable Vector Graphics Editor. Audio assets were made by Tiago Quendera using Audacity(R) except for the Wilhelm Scream (Wikimedia Commons). An example video (not from an experimental subject) showing the different environments is provided in the supplemental material. The task,

open-source code, and all assets are available at `https://github.com/quendera/human-foraging`.

Optogenetic stimulation

In order to optically stimulate ChR2 expressing VGAT-expressing GABAergic interneurons we used blue light from a 473 nm laser (LRS-0473-PFF-00800-03, Laserglow Technologies, Toronto, CA or DHOM-M-473-200, UltraLasers, Inc., Newmarket, CA) that was controlled by an acousto-optical modulator (AOM; MTS110-A1-VIS or MTS110-A3-VIS, AA optoelectronic, Orsay, FR) to deliver 10 ms pulses of light at 75 Hz, connected to Arduino Mega 2560 microcontroller board (Arduino, Somerville, US). Light exiting the AOM was focused into an optical fiber patch cord (200 µm, 0.22 NA, Doric lenses Inc, 357 rue Franquet, Quebec, Quebec, CA), connected to a second fiber patch cord through a rotary joint (FRJ 1x1, Doric lenses), which was then connected to the chronically implanted optic fiber cannula (MFC_200/230-0.48_3mm_ZF1.25(G)_FLT; Doric lenses Inc, 357 rue Franquet, Quebec, Quebec, CA). We estimated an average 15% loss of light power between the patch cord tip and the optic fiber cannula before surgery. In order to deliver light at 3 mW power, previously to each experiment day, the laser power at the tip of the patch cord was adjusted to 3.6 mW, to account for the estimated power loss. To test each protocol, we habituated animals to the new protocol for two days, then stimulated during six consecutive days. Stimulation was delivered on 50% of trials and started with the first valid nose-poke (that is to say after the infrared beam was broken for at least 100 ms). Stimulation ended if the animal did not nose-poke for 500 ms but would restart in case of another valid nose-poke on the same site.

Surgical procedures

Animals were anesthetized with isoflurane (4% induction and 0.5 - 1% for maintenance) and placed in a motorized computer-controlled Stoelting stereotaxic instrument with mouse brain atlas integration and real-time visualization of the surgery probe in the atlas space (Neurostar, Sindelfingen, Germany; `https://www.neurostar.de`). Antibiotic (Enrofloxacin, 2.5-5 mg/Kg, S.C.), pain killer (Buprenorphine,0.1 mg/Kg, S.C.), and local anesthesia over the scalp (0.2 mL, 2% Lidocaine, S.C.) were administered before incising the scalp. Target coordinates were 1.9 mm A.P., ±0.5 mm M.L., 1.75 mm D.V. for ACC and 2.9 mm A.P., ±1.25 mm M.L., 1.8 mm D.V. for OFC. Two craniotomies were performed above the target's coordinates for OFC implants. For ACC implants fiber were implanted over the target with an angle of ±16° on the ML axis to avoid damage to the superior sagittal sinus, and two craniotomies were performed at coordinates 1.9 mm A.P., ±1 mm M.L.

An optical fiber (200 µm core diameter, 0.48 NA, 510 mm) housed inside a connectorized implant (M3, Doric lenses, Quebec, Canada) was lowered into the brain (0 degree angle for OFC and 11 degree angle for ACC), through the craniotomy as the viral injection, and positioned 10 µm above the target. The implant was cemented to the skull using Super Bond C&B (Morita, Kyoto, Japan) and once dried covered with black dental cement acrylic (Pi-Ku-Plast HP 36, Bredent, Senden, Germany). The skin was stitched at the front and rear of the implant. Gentamicin(48760, Sigma-Aldrich, St. Louis, MO) was topically applied around the implant. Mice were monitored until recovery from the surgery and returned to their home cage where they were housed individually. Behavioral testing started at least 1 week after surgery to allow for recovery.

Histology and microscopy

For histological analysis mice were perfused transcardially with 4% paraformaldehyde (PFA) in phosphate buffer solution (PBS). After removing the brain they were left for 24 hours in 4% PFA solution in PBS, then transferred in 0.1% sodium azide solution in PBS. Brains were sliced in 50 µm coronal sections on a vibratome (Leica VT 1000 S), collected in wells maintaining the anterior-posterior order, and finally mounted on microscope slides (Thermo scientific, superfrost plus), with mowiol. Fluorescent images were acquired with an automated slide scanner (AxioScan Z1) equipped with a 10x, 0.45 NA PlanApochromat objective and a Hamamatsu OrcaFlash camera. Use of the appropriate filter combination allowed for DAPI and EYFP acquisition (Beam Splitter: 395, excitation: 330-375, emission: 430-470, and Beam splitter: 498, excitation: 453-485, emission: 507-546 respectively). Optic cannula placement was determined using coronal sections of the prefrontal cortex through which the fiber tract was visible. We determined the position by locating the section with the broadest base of the cannula tract and comparing the DAPI staining with the Allen Mouse Brain Atlas [75] (fig. S2; table S1).

2.4.5 Quantification and statistical analysis

Statistical analysis

The statistical analysis, which can be found in the Results section and figure legends, was performed using mixed effect models [85], in particular the Julia [17] implementation MixedModels.jl [13]. For each mixed model, we report the maximum likelihood estimate of the coefficient of interest $\pm$ the standard error of the estimate. Our N is the number of subjects: as different experiments had a potentially different number of subjects, we report it after every statistical test.

We fitted models with a random intercept (depending on subject identity) and compared nested models using a likelihood ratio test: in particular we used a chi-square test on the difference of the deviance of the two nested models, using as many degrees of freedom as the difference between the number of degrees of freedom of the two nested models [144]. That is to say, given two models m and n where n is a special case of m:

$$p = 1 - cdf\left(\chi^2_{dof(m)-dof(n)}, deviance(n) - deviance(m)\right).$$

When the p value is too small, we do not report the value but simply write $p < 1e-10$, which is floating point notation for $p < 10^{-10}$. To describe mixed models we will use Wilkinson notation (Wilkinson and Rogers, 1973), with $\mid$ denoting random effects and & denoting interaction terms. For example the formula:

$$ConsecutiveFailures \sim 1 + Protocol + Stimulation + (1 \mid MouseID)$$

uses as predictor for the number of consecutive failures after last reward a constant intercept, a coefficient for each protocol different than the medium protocol (which we consider as baseline), a coefficient for stimulation and a random intercept across mice. The formula

$$\begin{aligned} ConsecutiveFailures \sim & 1 + Protocol + Stimulation + \\ & Protocol\&Stimulation + (1 \mid MouseID) \end{aligned}$$

would also allow for an interaction term between protocol and stimulation. We did not test whether the data met the assumptions of the statistical methods used.

Task design

We designed a probabilistic foraging task for parallel use in mice and humans. Subjects sought rewards (water or points, respectively), by actively probing a foraging site (nose-poking or screen-tapping, respectively). Each site could be in one of two states, active or inactive. Each try in the active state yielded reward with probability p_{RWD}, and could cause the site to switch to the inactive state with probability p_{SW}. This required the subjects to travel to a second, fresh, site at some distance and bear a travel cost. Subjects were therefore tasked with inferring a hidden state (active or inactive) through a stochastic sequence of observations (rewards and failures).

Relevant statistics in the task

After a rewarded attempt, the subject could be sure to be in the correct location: ambiguity comes from failures, as it was possible that the target was correct but the subject was being unlucky. The more unsuccessful attempts, the higher the probability of a transition having occurred. Accumulated evidence in favor of a switch is a monotonically increasing function of the task parameters p_{RWD} and p_{SW}: the higher the reward probability, the more informative a failure is. Trivially, the higher the switch probability, the more likely the switch.

Possible task space representations

When analyzing our task, we consider two possible state representations. One is simpler and analogous to traditional approaches to modeling n-armed bandit tasks: the state corresponds to the current location of the subject (i.e., one of the two reward sites). The value of the two sites changes over time, yet the animals may be able to track this change with fast model-free learning. A second, more principled but more abstract

approach, postulates that the subjects tries to infer the optimal state representation, i.e., the probability that their current location is rewarding. In this model, there is no longer any need for fast online learning as the task representation is stable. The computation happening in real time is the inference process to compute this probability. To account for variability in the behavior, we allow both decision noise, distributed according to the soft-max rule, as well as inference noise (the inference process may be sub-optimal). We will refer to these two learning paradigms as stimulus-bound learning and inference-based learning respectively. It is important to note that, given the richer state representation, inference-based learning is the optimal way to solve the task and clearly outperforms simpler heuristics such as stimulus-bound learning.

Stimulus-bound learning

In stimulus-bound learning, we first define the relative value V as the difference of the value of the leftport and the value of the right port:

$$V = V_{LEFT} - V_{RIGHT}$$

We defined two auxiliary variables: a reward variable r indicating the outcome of each reward attempts, i.e., 1 for a reward and 0 for a failure, and a site variable s, indicating the current site, 1 for left and -1 for right.

We can define a signed outcome o of each reward attempt, which is:

$$o = r \cdot s,$$

that is to say, 1 for a reward on the left, -1 for a reward on the right and 0 for an omission.

For any attempt, we can then update the relative value using the signed outcome and some discount parameter γ

$$V_{t+1} = (1-\gamma)V_t + \gamma o_{t+1},$$

which admits an explicit solution:

$$V_t = (1-\gamma)^t \cdot V_0 + \gamma \cdot \sum_{i=1}^{t} (1-\gamma)^{t-i} \cdot o_i.$$

The probability of staying is a monotonically increasing function of the value of staying, so that rewards should make the animal more likely to stay and omissions more likely to leave, in a symmetric way.

Inference-based learning

We first derive recursive formulas to compute the probability that the current site is not rewarding as a function of the sequence of successful and failed reward attempts performed by the subject. In this model, the subject would compute the relative value as a function of the probability of the left (or right) site being active given the task history ($r_1, \ldots, r_t$ represent the outcomes of the various attempts and $s_1, \ldots, s_t$ the site of each attempt):

$$\begin{aligned} V_t = p_{RWD}(&P(LeftActive \mid r1, s_1, \ldots, r_t, s_t) - \\ &P(RightActive \mid r1, s_1, \ldots, r_t, s_t)). \end{aligned}$$

From value to decision

We have now defined to different ways to compute the relative value of left versus right, one directly based on reward accumulation, and one based on evidence accumulation. To define a behavior from this relative value,

we need to consider two more parameters. First of all we need a bias term T: as the two foraging sites are far apart, subjects should prefer to repeat site rather than alternate, to avoid the travel cost. Then we need a "inverse temperature" parameter β to describe how deterministic the animal is (with a very high β the animal would almost always choose the option with greater value, whereas with $\beta = 0$ the animal would choose randomly). We can then use the soft-max rule to generate behavior:

$$P(NextLeft \mid V, s) = \sigma(\beta(V + s \cdot T)),$$

where s represents the current site (1 for left and -1 for right).

In the simulations we will use the same softmax rule to simulate behavior: the difference between stimulus-bound and inference-based learning derives from the different procedures used to compute the relative value.

Computing the likelihood ratio

In inference-based learning we defined the relative value as a function of the relative difference:

$$P(LeftActive \mid r1, s_1, \ldots, r_t, s_t) - P(RightActive \mid r1, s_1, \ldots, r_t, s_t).$$

This quantity can be computed recursively. To do so we will need an auxiliary variable. We define R_t as the probability ratio that the current site is active or inactive given task history:

$$R_t = \frac{P(Inactive \mid r1, s_1, \ldots, r_t, s_t)}{P(Active \mid r1, s_1, \ldots, r_t, s_t)}$$

From R_t we can compute V_t as follows:

$$V_t = p_{RWD} \cdot \left(1 - \frac{2}{R_t^{-s_t} + 1}\right).$$

Rather than computing V_t recursively directly, we notice that R_t respects a simple recursive equation (likelihood ratio update equation):

$$R_{t+1} = \left(\frac{R_t + p_{SW}}{1 - p_{SW}}\right)^{s_t s_{t+1}} \cdot \frac{P(r_{t+1} \mid Inactive)}{P(r_{t+1} \mid Active)}$$

where p_{SW} represents the probability of switching from active to inactive state.

The term $\frac{R_t + p_{SW}}{1 - p_{SW}}$ is the ratio between the following two equations:

$$P(NextInactive) = P(Inactive) + p_{SW} P(Active)$$
$$P(NextActive) = (1 - p_{SW}) P(Active)$$

The exponent $s_t s_{t+1}$ in section 2.4.5 simply means that the probability ratio $\frac{P(Inactive)}{P(Active)}$ inverts when the subject changes site. Finally the term $\frac{P(r_{t+1} \mid Inactive)}{P(r_{t+1} \mid Active)}$ represents the new evidence acquired with the outcome of attempt $t + 1$. In the case of a reward, $P(r_{t+1} \mid Inactive) = 0$, so:

$$R_{t+1} = 0$$

If r_{t+1} is a failure, then $P(r_{t+1} \mid Inactive) = 1$ whereas $P(r_{t+1} \mid Active) = 1 - p_{RWD}$, so the likelihood ratio update equation simplifies to:

$$R_{t+1} = \left(\frac{R_t + p_{SW}}{1 - p_{SW}}\right)^{s_t s_{t+1}} \cdot \frac{1}{1 - p_{RWD}}.$$

Having established that R_t resets to 0 with a reward, we can analyze the most interesting case for a probability computation: a sequence of attempts on the same site (let us say $s_1, \ldots, s_t = 1$) where the first attempt is rewarded (thus resetting the probability) and the following are not.

As the first attempt is rewarded, $R_1 = 0$. Furthermore, if we assume that all attempts are on the same site, the likelihood ratio grows following

the recursive equation:

$$R_{t+1} = \frac{R_t + p_{SW}}{(1 - p_{SW})(1 - p_{RWD})} > R_t.$$

We can define the auxiliary quantity

$$\varrho = \frac{1}{(1 - p_{SW})(1 - p_{RWD})}.$$

Our recursive equation becomes:

$$R_{t+1} = \varrho \cdot (R_t + p_{SW}).$$

This is a standard linear recursion that we can solve with a linear transformation

$$S_t = R_t + \frac{p_{SW}}{\varrho - 1}.$$

The recursion of S_t is:

$$S_1 = \frac{p_{SW}}{\varrho - 1} \qquad \text{and} \qquad S_{t+1} = \varrho \cdot S_t,$$

whose solution is

$$S_t = p_{SW} \frac{\varrho^{t-1}}{\varrho - 1},$$

therefore:

$$R_t = p_{SW} \frac{\varrho^{t-1} - 1}{\varrho - 1}.$$

That is to say R_t grows exponentially with rate $\log(\varrho) = -\log(1 - p_{SW}) - \log(1 - p_{RWD})$. Increasing either p_{RWD} or p_{SW} would increase the growth rate of R.

2.4.6 Data and code availability

All analysis was performed using custom code written in Julia [17]. The code used to simulate value or inference models is available on GitHub, under the MIT license, at `https://github.com/piever/ValueInferenceTools.jl`. The data is published on Zenodo with DOI 10.5281/zenodo.3607558 and can be found at `https://zenodo.org/record/3607558`.

2.5 Author contributions

P.V., E.L., D.S., and Z.F.M. designed the experiments and analyses. P.V., D.S., B.G., I.T., T.Q., and M.N.O.L. conducted the experiments. P.V. and E.L. analyzed the data. P.V. wrote the original draft. E.L., D.S., and Z.F.M. reviewed and edited the paper.

Chapter 3

Parametric machines

Summary

Using tools from category theory, we provide a framework where artificial neural networks, and their architectures, can be formally described. We first define the notion of *machine* in a general categorical context, and show how simple machines can be combined into more complex ones. We explore finite- and infinite-depth machines, which generalize neural networks and neural ordinary differential equations. Borrowing ideas from functional analysis and kernel methods, we build complete, normed, infinite-dimensional spaces of machines, and discuss how to find optimal architectures and parameters—within those spaces—to solve a given computational problem. In our numerical experiments, these kernel-inspired networks can outperform classical neural networks when the training dataset is small.

3.1 Introduction

Background. In recent years, the deep learning framework has achieved and surpassed state-of-the-art performance in many machine learning

tasks, using a variety of architectures. Notably, in the field of computer-vision, Convolutional Neural Networks (CNNs) showcase impressive performance [72]. However, a paradoxical problem affects the performance and robustness of deep neural networks. Deeper networks should in principle perform at least as well as shallower ones, finding in the limit of infinite layers a solution where the extra layers approximate the identity function. However, [55] reports that deeper architectures can cause a degradation of performance not explained by overfitting. Choosing a deep architecture is therefore a difficult task, where one needs to rely on heuristics, or brute trial and error. Current approaches to automated architecture search [43] rely on large or augmented training datasets and manually engineered building-blocks. Moreover, they often lack principled regularization methods and guarantees of optimality.

Aim and contributions. We propose a theoretical framework where neural networks can be formally described as a special case of a more general construction—*parametric machines*. Using the language of category theory, we introduce this notion in a variety of settings. Modularity—a fundamental property of standard neural architectures—is intrinsic to this construction: it is possible to create complex machines as a sum of simpler ones. Our notion unifies seemingly disparate architectures, ranging from hand-designed combinations of layers, graphically represented here via a hypergraph, to networks defined via differential equations [32]. The key intuition is that a neural network can be considered as an endomorphism f on a space of global functions (defined on all neurons on all layers). If such a network is feedforward, then $\mathrm{id} - f$ is invertible, and its inverse can be computed via a forward pass. The two broad classes of architectures that we describe here are the analogous of the classical results that $\mathrm{id} - f$ is invertible if f is a linear nilpotent map (finite depth) or a contraction (infinite depth). Our ambition is to define architectures with little

or no human intervention. Infinite-depth machines generalize neural ordinary differential equations, by adding a choice of architecture. Unlike the finite-depth case, whose structure can be represented by a hypergraph, this architecture is defined in terms of continuous functions and, therefore, can be parameterized and optimized during training. When the training dataset is small, we rely on kernel methods to guarantee optimality. Finite- and infinite-depth *kernel machines* exhibit *all* shortcut connections, thus avoiding pathologies due to the architecture depth. Such dense connectivity does not cause a quadratic increase in the number of parameters in the case of small datasets. In addition to the theoretical framework, we test our main algorithms, namely *hypergraph neural architecture search*, and *discrete* and *continuous kernel machines*, in three applications, proving their effectiveness, with a focus on small datasets. Each algorithm has been wrapped as a PyTorch [96] module, and can be used both as standalone or *layer* of a classical neural network architecture.

Structure. Section 3.2.1 discusses the necessary categorical preliminaries. Building on those, we introduce the notion of *machine* and its *stable state*. These generalize the connection between global nonlinear operators on function spaces and the forward pass of a layered neural network or neural Ordinary Differential Equation (ODE), see section 3.2.2 respectively. In section 3.2.3, taking advantage of the framework developed in sections 3.2.1 and 3.2.2, we define a novel architecture based on operator-valued kernels and filtrations of Hilbert spaces. The proposed constructions are tested on different tasks and compared with state-of-the-art methods.

3.2 Results

3.2.1 Machines

We lay our fundamental definitions in arbitrary categories, i.e. collections of objects (such as vector spaces or topological spaces) and morphisms (such as linear or continuous functions) between them. The basic ingredient is a functor ι from a *linear* category $\mathbf{L}$ to an arbitrary category $\mathbf{C}$, that is to say a mapping that associates to each object $M \in \mathrm{Obj}(\mathbf{L})$ an object $\iota(M) \in \mathrm{Obj}(\mathbf{C})$, and to each morphism $m\colon M_1 \to M_2$ a morphism $\iota(m)\colon \iota(M_1) \to \iota(M_2)$, compatible with identity and composition.

This grants us some additional flexibility: the source category $\mathbf{L}$ could have different levels of structure, adapting to different data types and giving the possibility to manually incorporate previous knowledge about the data [16]. The choice of the target category is interesting for two reasons. On the one hand, it constrains the machine, depending on what morphisms are allowed in $\mathbf{C}$ (e.g., all continuous functions, only Lipschitz functions, smooth functions). On the other hand, it adds structure to the machine, allowing, for example, continuous or smooth parameterizations.

Categorical preliminaries

Let R be a commutative ring. A R-linear category (or R-algebroid) is a category $\mathbf{L}$ such that, for $L, M \in \mathrm{Obj}(\mathbf{L})$, $\mathrm{Hom}_{\mathbf{L}}(L, M)$ is a R-module, and composition of morphisms is bilinear. We require that both $\mathbf{L}$ and $\mathbf{C}$ have finite products, and that the functor ι preserves them. In R-linear categories, finite products coincide with finite coproducts: they are generally denoted by $\oplus$.

Let M be an object in $\mathbf{L}$, X an object in $\mathbf{C}$, and $\iota\colon \mathbf{L} \to \mathbf{C}$. Then $\mathrm{Hom}_{\mathbf{C}}(X, \iota(M))$ is a R-module. Indeed, given $f, g\colon X \to \iota(M)$, we can define

$$X \xrightarrow{f \times g} \iota(M) \times \iota(M) \simeq \iota(M \oplus M) \xrightarrow{\iota(+)} \iota(M).$$

To simplify the notation, we will denote the resulting morphism $f + g$. Given $\lambda \in R$, and $f\colon X \to \iota(M)$, we can consider the morphism $\iota(\lambda \cdot \mathrm{id})f$, which we denote λf for simplicity. Note that, with this definition, it is always true that $(\lambda f + \mu g) \circ h = \lambda f \circ h + \mu g \circ h$, but not necessarily $h \circ (\lambda f + \mu g) = \lambda h \circ f + \mu h \circ g$. Composition is therefore only linear in one argument in $\mathrm{End}_{\mathbf{C}}(\iota(M))$. However, the latter equality holds whenever $h = \iota\left(\tilde{h}\right)$, for some $\tilde{h} \in \mathrm{End}_{\mathbf{L}}(M)$.

As a possible example of such pairs of categories, the reader can consider **Ban**, the category of complete normed vector spaces (Banach spaces) with bounded (hence, continuous) linear functions as morphisms, and **Top**, the category of topological spaces and continuous functions. The forgetful functor $\mathrm{forget}\colon \mathbf{Ban} \to \mathbf{Top}$, associating to each Banach space its underlying topological space preserves finite products. Indeed, both **Ban** and **Top** have finite products, denoted $\oplus$ and $\times$ respectively, and they are compatible with forget: $\mathrm{forget}(A \oplus B)$ is canonically isomorphic to $\mathrm{forget}(A) \times \mathrm{forget}(B)$.

The concrete case $\mathbf{L} = \mathbf{Ban}$ and $\mathbf{C} = \mathbf{Top}$ does not require knowledge of category theory to be understood. Readers unfamiliar with category theory may replace *morphism in* $\mathbf{L}$ with *linear, continuous function*, and *morphism in* $\mathbf{C}$ with *continuous function* (non necessarily linear).

Parameterized morphisms. In order to allow parameterizations in our framework, we will use the classical construction of *Kleisli category* [82, Chapt. VI] over the *product comonad* [136]. As the general construction on an arbitrary comonad is quite abstract, we describe it concretely in this case. Given an object P—the parameter space—in a category $\mathbf{C}$ with finite products, we can build a new category $\mathbf{C}_P$, which has the same objects as $\mathbf{C}$ and such that

$$\mathrm{Hom}_{\mathbf{C}_P}(X, Y) := \mathrm{Hom}_{\mathbf{C}}(P \times X, Y).$$

$\mathbf{C}_P$ can be shown to be a category with the obvious composition. Intuitively, morphisms in $\mathbf{C}_P$ can be thought of as morphisms in $\mathbf{C}$ parameterized by P. We have a functor $\mathbf{C} \to \mathbf{C}_P$ given by the identity on objects and on morphisms

$$\begin{aligned} \mathrm{Hom}_{\mathbf{C}}(X, Y) \to \mathrm{Hom}_{\mathbf{C}_P}(X, Y) &= \mathrm{Hom}_{\mathbf{C}}(P \times X, Y) \\ m &\mapsto \pi_X \circ m. \end{aligned}$$

It is straightforward to verify that this functor $\mathbf{C} \to \mathbf{C}_P$, preserves products. Hence, whenever we have a functor that preserves finite products $\iota \colon \mathbf{L} \to \mathbf{C}$, for each $P \in \mathrm{Obj}(\mathbf{C})$ we can define by composition a corresponding

$$\iota_P \colon \mathbf{L} \to \mathbf{C}_P, \tag{3.1}$$

which also preserves finite products.

Stable state

The definitions given in the previous section allow for the creation of a framework in which complex architectures can be formally described. We start by describing how, in the classical deep learning framework, different layers are combined to form a network. Intuitively, function composition seems the natural operation to do so. A sequence of layers

$$X_0 \xrightarrow{l_1} X_1 \xrightarrow{l_2} \dots X_{n-1} \xrightarrow{l_n} X_n$$

is composed into a map $X_0 \to X_n$. However, this intuition breaks down in the case of shortcut connections or more complex, non-sequential architectures.

From a mathematical perspective, a natural alternative is to consider a *global* space $X = \bigoplus_{i=0}^{n} X_i$, and the global endofunction

$$f = \sum_{i=1}^{n} l_i \colon X \to X.$$

What remains to be understood is the relationship between the function f and the layer composition $l_n \circ l_{n-1} \circ \cdots \circ l_2 \circ l_1$. To clarify this relationship, we assume that the output of the network is the entire space X, and not only the output of the last layer, X_n. Let the input function be the inclusion $g \colon X_0 \to X$. The network transforms g into a map $h \colon X_0 \to X$, induced by $l_i \circ \cdots \circ l_1 \colon X_0 \to X_i$, for $i \in \{0, \ldots, n\}$. From a practical perspective, h computes the activation values of all the layers and stores not only the final result, but also all the activations of the intermediate layers.

The key observation, on which our framework is based, is that f and g alone are sufficient to determine h. Indeed, h is the only map $X_0 \to X$ that respects the following property:

$$h = g + fh. \tag{3.2}$$

Equation (3.2) holds also in the presence of shortcut connections, or more complex architectures such as UNet [77] (see section 3.2.2). The existence of a unique solution to eq. (3.2) for any choice of input function g will be the defining property of a *machine*, our generalization of a feedforward deep neural network.

Definition 1. *Let R be a commutative ring, and $\mathbf{L}$ a R-linear category. Let $\iota \colon \mathbf{L} \to \mathbf{C}$ be a functor that preserves finite products, and $M \in \mathrm{Obj}(\mathbf{L})$. An endomorphism $f \in \mathrm{End}_{\mathbf{C}}(\iota(M))$ is a* machine *if, for all morphisms*

$g\colon X \to \iota(M)$, there exists a unique $h\colon X \to \iota(M)$ such that:

$$h = g + fh.$$

We call h the stable state *of f with initial condition g, and denote by S_f the stable state of f with initial condition* $\mathrm{id}_{\iota(M)}$.

The following result will be crucial to compute stable states in the remainder of this work.

Theorem 1. *$f \in \mathrm{End}_{\mathbf{C}}(\iota(M))$ is a machine if and only if $\mathrm{id} - f$ is an isomorphism. Whenever that is the case, the stable state with initial condition g is given by $(\mathrm{id} - f)^{-1} \circ g$. In particular, the stable state of f is $S_f = (\mathrm{id} - f)^{-1}$.*

Proof. Let us assume that f is a machine. $S_f = \mathrm{id} + fS_f$, so $(\mathrm{id} - f)S_f = \mathrm{id}$, hence $\mathrm{id} - f$ is a split epimorphism. Let h, h' be such that $(\mathrm{id} - f)h = (\mathrm{id} - f)h'$. Then both h and h' are stable states of f with initial condition $(\mathrm{id} - f)h$, hence they must be equal, so $\mathrm{id} - f$ is monic. A monic split epimorphism is necessarily an isomorphism. Conversely, let us assume that $\mathrm{id} - f$ is an isomorphism. Then $h = g + fh$ if and only if $h = (\mathrm{id} - f)^{-1}g$. □

Parametric machines. The specific choice of categories and functors in definition 1 determines the nature of the machine. We can incorporate the notion of *parameter space* as follows. Given a functor $\iota\colon \mathbf{L} \to \mathbf{C}$, such as forget$\colon \mathbf{Ban} \to \mathbf{Top}$, and an object $P \in \mathbf{C}$, we can construct a novel functor ι_P given by the composition

$$\iota_P\colon \mathbf{L} \to \mathbf{C}_P,$$

as in eq. (3.1). If ι preserves finite products, then so does ι_P (see section 3.2.1). A machine with respect to the functor ι_P is automatically equipped with a parameterization based on the space P that can be used

in optimization. We call such machine a *parametric machine*, with *parameter space* P.

Convergence and depth

All nilpotent linear endomorphisms of a Banach space are machines. Continuous endofunctions with norm strictly smaller than 1 (i.e., not necessarily linear contractions) are also machines. In both cases, the stable state can be found by considering the following sequence:

$$h_0 = \mathrm{id} \qquad \text{and} \qquad h_{n+1} = \mathrm{id} + f h_n. \tag{3.3}$$

Even though for different reasons, both in the nilpotent, linear case and in the contraction case, $\|h_m - h_n\|$ converges to 0 for sufficiently large m, n. If f is nilpotent and linear, then $h_{n+1} - h_n = f(h_n - h_{n-1})$, so it will go to 0 in a finite number of steps. If instead f has norm $\lambda < 1$, then

$$\|h_{n+1} - h_n\| = \|f h_n - f h_{n-1}\| \leq \lambda \|h_n - h_{n-1}\|.$$

Therefore, consecutive distances are uniformly bounded by $c\lambda^n$ for some c, hence, for $m \geq n$, $\|h_m - h_n\| \leq \frac{c\lambda^n}{1-\lambda}$, thus ensuring convergence.

Definition 2. *Let $f, \{h_i\}_{i \in \mathbb{N}}$ be as in eq.* (3.3)*. The* depth *of f is the smallest integer n (if it exists) such that*

$$h_{n+1} = h_n,$$

and ∞ otherwise.

Modularity and computability

Under suitable *independence* conditions, more complex machines can be created as a sum of simpler ones.

Definition 3. *Let $R, \mathbf{C}, \mathbf{L}, \iota, M$ be as in definition 1. Let $f, f' \in \mathrm{End}_{\mathbf{C}}(\iota(M))$. We say that f* does not depend *on f' if, for any $X \in \mathrm{Obj}(\mathbf{C})$, for any pair of maps $b, b' \colon X \to \iota(M)$, and for all $\lambda \in R$, the following holds:*

$$f(b + \lambda f' b') = fb. \tag{3.4}$$

Otherwise, we say that f depends on f'.

Remark 1. *Independence of f from f' is stronger than asking $ff' = 0$, because in general it is not true that $f(a + a') = fa + fa'$.*

Definition 3 is quite useful to compute stable states. For example, if f does not depend on itself, then automatically f is a machine, and $S_f = \mathrm{id} + f$; we call such machines *square-zero*. Under suitable assumptions, machines can be juxtaposed to recover the notion of *deep neural networks*.

Theorem 2. *Let f, f' be machines such that f does not depend on f'. Then $f + f'$ is also a machine, and $S_{f+f'} = S_{f'} S_f$. If furthermore f' does not depend on f, then $S_{f+f'} = S_f + S_{f'} - \mathrm{id}$.*

Proof. By theorem 1 and eq. (3.4), $f + f'$ is a machine:

$$(\mathrm{id} - f)(\mathrm{id} - f') = (\mathrm{id} - f - f'), \tag{3.5}$$

so $(\mathrm{id} - f - f')$ is an isomorphism (composition of isomorphisms). Equation (3.5) also determines the stable state:

$$S_{f+f'} = (\mathrm{id} - f - f')^{-1} = (\mathrm{id} - f')^{-1}(\mathrm{id} - f)^{-1} = S_{f'} S_f.$$

Moreover, if f' does not depend on f, then

$$f(S_f + S_{f'} - \mathrm{id}) = f(S_f + f' S_{f'}) = f S_f,$$
$$f'(S_f + S_{f'} - \mathrm{id}) = f'(f S_f + S_{f'}) = f' S'_f.$$

Hence,

$$S_f + S_{f'} - \mathrm{id} = \mathrm{id} + (f + f')(S_f + S_{f'} - \mathrm{id}).$$

□

Theorem 2 allows us to build a broad class of networks from basic components. Given a set of machines $\{f_1, \ldots, f_n\}$, we can define its *dependency graph* as follows: the set of vertices is $\{1, \ldots, n\}$, and there is a directed edge from i to j (for $i \neq j$) if and only if f_j depends on f_i. If the dependency graph is acyclic, then $f_1 + \cdots + f_n$ is a machine, and there is an efficient procedure to compute its stable state. We will need some basic graph-theoretical notions to describe it.

Layering of acyclic directed graphs. Given a finite directed graph (V, E), a *layering* [128] on (V, E) of height k is a partition $\{V_1, \ldots, V_k\}$ of its vertices such that, whenever we have an edge from $v_i \in V_i$ to $v_j \in V_j$, then necessarily $i < j$. A directed graph (V, E) can only admit layerings if it is acyclic. In that case, the height of a layering must be at least the length of the longest path in (V, E) increased by one. This lower bound is tight. Indeed, given a vertex $v \in V$, we can define its *depth* $d(v)$ to be the length of the longest path terminating in v. A layering of minimal height can be defined as follows:

$$V_i = d^{-1}(i + 1).$$

Corollary 1. *Let us consider a set of machines $\{f_1, \ldots, f_n\}$, and let (V, E) be its dependency graph. Let us assume that (V, E) is acyclic, with layering $\{V_1, \ldots, V_k\}$. If we denote $f = f_1 + \cdots + f_n$, then*

$$S_f = \left(\mathrm{id} + \sum_{v \in V_k} (S_{f_v} - \mathrm{id})\right) \ldots \left(\mathrm{id} + \sum_{v \in V_1} (S_{f_v} - \mathrm{id})\right). \tag{3.6}$$

Proof. We can define a new set of machines $\{l_1, \dots, l_k\}$, where

$$l_i = \sum_{v \in V_i} f_v.$$

For $i < j$, l_i does not depend on l_j, so eq. (3.6) follows trivially from theorem 2:

$$S_f = S_{l_k} \dots S_{l_1} = \left(\mathrm{id} + \sum_{v \in V_k} (S_{f_v} - \mathrm{id}) \right) \dots \left(\mathrm{id} + \sum_{v \in V_1} (S_{f_v} - \mathrm{id}) \right).$$

□

Corollary 1 establishes a clear link between sums of independent machines and compositions of layers in classical feedforward neural networks. Even though, in general, we are not limited to sequential architectures (see fig. 3.2), the layering procedure determines the order in which machines should be concatenated.

3.2.2 Finite and infinite depth

Neural networks can be seen as a sum of independent square-zero machines, one per layer. We first use our machine-based framework to design finite-depth architectures using directed hypergraphs. This allows for shortcut connections [18, 105], as in, for instance, residual learning networks [55], as well as more complex connectivities, such as UNet [77].

Analogously, ODEs correspond to a sum of independent contracting machines, obtained by splitting the time interval into small sub-intervals. This is a standard strategy to obtain existence and uniqueness results for ODEs, which are a consequence of the Caccioppoli-Banach principle [65, Chapt. XVI]—contractions in a complete metric space admit a unique fixed point. As described in section 3.2.1, unlike square-zero machines, which

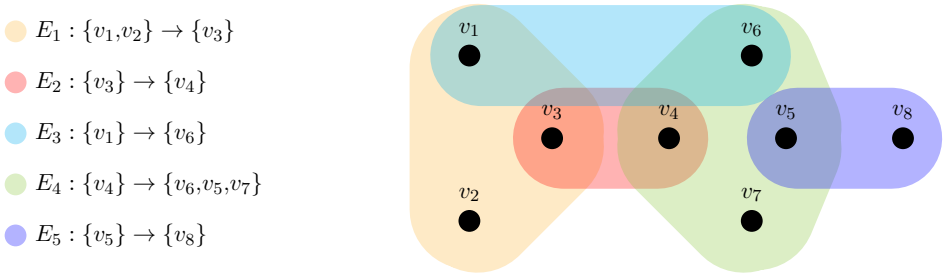


Figure 3.1: **Hypergraph representation of a neural network.** Given layers $\{l_1, \ldots, l_5\}$, the representation corresponds to the neural network mapping $(x_1,\ x_2,\ x_3,\ x_4,\ \ldots, x_8)$ to $(x_1,\ x_2,\ l_1(x_1, x_2) + x_3,\ l_2(l_1(x_1, x_2) + x_3) + x_4,\ \ldots,\ l_5(l_4(l_2(l_1(x_1, x_2) + x_3) + x_4) + x_5) + x_8)$.

have depth 1, contracting machines can in general have infinite depth. We describe *Volterra machines*, a generalization of neural ODEs [32] in our framework, as an example of an infinite-depth machine.

Hypergraph machines

We will need some basic notions concerning directed hypergraphs from [47].

Definition 4. *[47, Sect. 2] Let* $\mathcal{P} : \mathbf{Set} \to \mathbf{Set}$ *denote the power set functor. A directed hypergraph is a pair of finite sets* $(\mathcal{V}, \mathcal{E})$ *of vertices and hyperedges, with* $\mathcal{E} \subseteq \mathcal{P}\mathcal{V} \times \mathcal{P}\mathcal{V}$*, that is to say each hyperedge* E *can have several source vertices (or none) and several target vertices (or none). We denote the subset of source vertices and target vertices* $s(E)$ *and* $t(E)$ *respectively. In the remainder of this work, directed hypergraphs will simply be called hypergraphs.*

Even though [47] requires hyperedges to have disjoint source and target, we drop this condition. The notion of acyclic hypergraph is identical as hyperedges with overlapping source and target are cycles of length 1.

Definition 5. *[47, Sect. 3] Given a hypergraph $(\mathcal{V}, \mathcal{E})$, a* path P_{ab} *of length q is a sequence $v_1 = a, E_1, v_2, E_2, \ldots, E_q, v_{q+1} = b$, where:*

$$a \in s(E_1), \quad v_j \in t(E_{j-1}) \cap s(E_j),\ j \in \{2, \ldots, q\}, \quad \text{and} \quad b \in t(E_q).$$

P_{ab} is a cycle *if $b \in s(E_1)$. A hypergraph is* acyclic *if it has no cycles.*

Definition 6. *The* line *graph of a directed hypergraph $H = (\mathcal{V}, \mathcal{E})$ is a directed graph having as nodes the set $\mathcal{E}$ of hyperedges of H. E_1 is connected to E_2 if and only if $t(E_1) \cap s(E_2) \neq \emptyset$.*

Let $(\mathcal{V}, \mathcal{E})$ be an acyclic hypergraph. A *nonlinear hypergraph representation* is, for each vertex $v \in \mathcal{V}$, an object $M_v \in \mathrm{Obj}(\mathbf{L})$, and, for each hyperedge $E \in \mathcal{E}$, a map:

$$\iota\left(\bigoplus_{v \in s(E)} M_v\right) \xrightarrow{p_E} \iota\left(\bigoplus_{v \in t(E)} M_v\right).$$

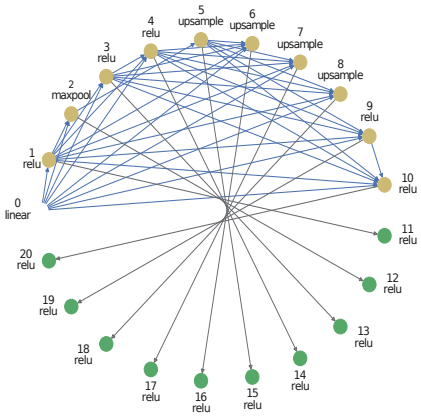
Let $M := \bigoplus_{v \in \mathcal{V}} M_v$. Then p_E can be extended to a machine on M:

$$\iota(M) \longrightarrow \iota\left(\bigoplus_{v \in s(E)} M_v\right) \xrightarrow{p_E} \iota\left(\bigoplus_{v \in t(E)} M_v\right) \longrightarrow \iota(M)$$

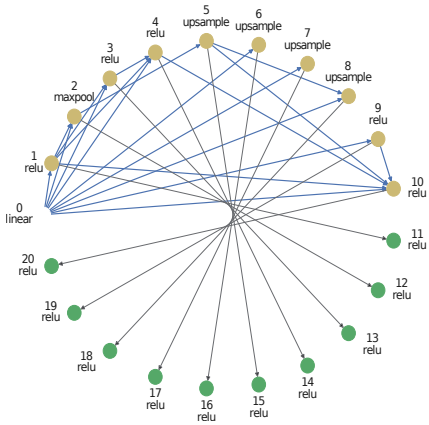
The dependency graph for $\{p_E\}_{E \in \mathcal{E}}$ is a subgraph of the line graph of $(\mathcal{V}, \mathcal{E})$, and is therefore also acyclic, hence the endomorphism $\sum_{E \in \mathcal{E}} p_E$ is a machine.

Prunable directed graph architectures. Using the graph-theoretical ideas developed so far, we devised a simple architecture search algorithm that requires minimal fine-tuning. We start with a finite number of nodes, each equipped with an activation function on a given space with a group of symmetries (i.e., translations for convolution, identity for fully-connected layers). Each node is connected to all preceding nodes with compatible dimensionality and has a unique fully-connected output. When reaching a

(a) Prunable hypergraph, initialization.



(b) Trained hypergraph, MNIST.



(c) Learned convolutional architecture.

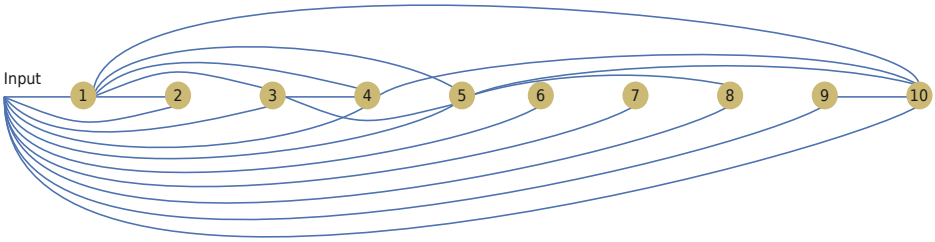


Figure 3.2: As a starting architecture we consider a directed acyclic graph whose nodes are activation functions (identity, ReLU, upsampling and max-pooling). Blue directed edges are convolutions and black directed edges are linear layers. In a we show the starting connectivity, which is maximal with respect to the blue edges which connect all admissible (i.e. same dimensionality) activation nodes. During training, we prune those edges whose weights have sufficiently small Euclidean norm. b Architecture after pruning during training on the MNIST dataset (with accuracy $\approx 98.6\%$). c The learned convolutional architecture.

node, the outputs of its incoming edges are summed. During training, we add to the loss function a cost proportional to the sum of the Euclidean norms of the weights associated with each edge. In fig. 3.2a we show this construction for a translation-equivariant architecture used to classify the MNIST dataset [39], where we start with 10 nodes equipped with activation functions compatible with an image analysis task, connected by convolutional edges with a fixed number of channels. During training, we prune edges whose associated weights have Euclidean norm smaller than a fixed tolerance (10^{-6}), see fig. 3.2b. This small tolerance value has minimal impact on the accuracy of the model while reducing its computational cost. In fig. 3.2c, we observe that the learned convolutional architecture has non-trivial connectivity. The achieved accuracy on the MNIST test set ($\approx$ 98.6%) is below state of the art. However, this particular algorithm does not require any manual fine-tuning, other than the choice of equivariance and number and dimension of nodes, which could be chosen automatically according to the computational power of the user's machine.

Volterra machines

A natural generalization of neural ODEs in our framework is given by *Volterra machines*. The nonlinear *Volterra equation of the second kind* is, in its classical form:

$$u(t) = \psi(t) + \int_{t_0}^{t} \phi(t, s, u(s))ds, \text{ for all } t \in [t_0, T], \tag{3.7}$$

where $t_0 < T \in \mathbb{R}$. This equation generalizes ordinary differential equations. Whenever ϕ only depends on the last two arguments, i.e. $\phi(t, s, v) = \phi(s, v)$, and $\psi(t) = \psi(t_0)$ for all $t \in [t_0, T]$, then the solution u of the Volterra equation (if it exists) also solves the initial value problem:

$$\frac{du(t)}{dt} = \phi(t, u(t)) \text{ and } u(t_0) = \psi(t_0).$$

We consider the vector-valued case, where the codomain of ϕ, ψ (and consequently u) is the finite-dimensional Hilbert space $\mathbb{R}^n$, equipped with the standard scalar product. Let $L^2([t_0, T], n)$ be the Hilbert space of square-integrable functions from the interval $[t_0, T]$ to $\mathbb{R}^n$. We deviate slightly from the more standard set of assumptions (see [11]) to ensure existence and uniqueness of solutions, as we do not ask that ψ is continuous:

1. $\psi \in L^2([t_0, T], n)$.
2. $\phi(t, s, v)$ is continuous for $t_0 \leq s \leq t \leq T$.
3. $\phi(t, s, v)$ satisfies a uniform Lipschitz condition in v for $t_0 \leq s \leq t \leq T$. That is to say, there exists $\lambda \in \mathbb{R}$ such that, for all $v, \tilde{v} \in \mathbb{R}^n$,

$$\|\phi(t, s, v) - \phi(t, s, \tilde{v})\| \leq \lambda \|v - \tilde{v}\|. \tag{3.8}$$

Using the machine framework, we show existence and uniqueness of solutions for square-integrable functions.

Definition 7. *Let $\phi(t, s, v)$ be a continuous function on $t_0 \leq s \leq t \leq T$ and $v \in \mathbb{R}^n$, with values in $\mathbb{R}^n$. If $\phi(t, s, v)$ satisfies a uniform Lipschitz condition in v for $t_0 \leq s \leq t \leq T$, we say that ϕ is a* Volterra machine *on $L^2([t_0, T], n)$.*

A Volterra machine ϕ is a machine on $L^2([t_0, T], n)$. Let

$$f \in \mathrm{End}_{\mathbf{Top}}(L^2([t_0, T], n))$$

be the nonlinear endofunction given by:

$$f(u) = t \mapsto \int_{t_0}^{t} \phi(t, s, u(s)) ds.$$

Let λ be such that eq. (3.8) holds. Let us choose a positive integer N such that

$$N > \lambda^2 (T - t_0)^2. \tag{3.9}$$

For $i \in \{0, \ldots, N\}$, let $t_i = t_0 + \frac{i}{N}(T - t_0)$. For $i \in \{1, \ldots, N\}$, we can define

$$f_i(u) = t \mapsto \int_{t_0}^{t} \phi(t, s, u(s)) \mathbb{1}_{[t_{i-1}, t_i]}(s) ds.$$

Clearly $f = f_1 + \cdots + f_N$. Furthermore, for $i < j$, f_i does not depend on f_j. We need to show that f_i is a contraction. Then, given $u, \tilde{u} \in L^2([t_0, T], n)$, we have:

$$\begin{aligned}
\|f_i(u) - f_i(\tilde{u})\|_2^2 &= \int_{t_0}^{T} \left\| \int_{t_0}^{t} \mathbb{1}_{[t_{i-1}, t_i]}(s) [\phi(t, s, u(s)) - \phi(t, s, \tilde{u}(s)] ds \right\|_2^2 dt \\
&\leq \int_{t_0}^{T} \frac{T - t_0}{N} \int_{t_0}^{t} \|\phi(t, s, u(s)) - \phi(t, s, \tilde{u}(s))\|_2^2 \, ds dt \\
&\leq \int_{t_0}^{T} \frac{T - t_0}{N} \int_{t_0}^{t} \lambda^2 \|u(s) - \tilde{u}(s)\|_2^2 \, ds dt \\
&\leq \int_{t_0}^{T} \lambda^2 \frac{T - t_0}{N} \|u - \tilde{u}\|_2^2 \, dt \\
&= \lambda^2 \frac{(T - t_0)^2}{N} \|u - \tilde{u}\|_2^2
\end{aligned}$$

Therefore, by eq. (3.9), f_i is a contraction. As f is a sum of machines with an acyclic dependency graph, it is also a machine on $L^2([t_0, T], n)$ by corollary 1. In particular, given a sequence $\{\psi_n\}_{n \in \mathbb{N}} \to \psi_\infty$ of square-integrable functions that converges in norm L^2 to ψ_∞, for all $n \in \mathbb{N} \cup \infty$ there is a unique u_n such that

$$u_n(t) = \psi_n(t) + \int_{t_0}^{t} \phi(t, s, u_n(s)) ds, \text{ for all } t \in [t_0, T],$$

and the sequence $\{u_n\}_{n \in \mathbb{N}}$ converges in norm L^2 to u_∞.

Efficient Volterra machines

Nonlinear Volterra integral equations are in general harder to solve than ordinary differential equations (see [11] for a review of possible methods). This is particularly problematic here, as we wish to solve a Volterra equation in a time comparable with the forward pass of a neural ODE. Luckily, some special cases of Volterra equations admit a simpler solution in terms of a system of ODEs [21]. Let U, V, W be finite real vector spaces equipped with a bilinear map $B\colon U \otimes V \to W$. Let $\phi_1, \ldots, \phi_m$ be U-valued functions, and $c_1, \ldots, c_m$ V-valued functions. We can consider:

$$\phi(t, s, v) = \sum_{j=1}^{m} B(\phi_j(s, v), c_j(t)).$$

Analogously to a result presented in [21], we can solve the corresponding Volterra equation as a system of ODEs.

Theorem 3. *[21, Thm. 3] Let $\psi \in L^2([t_0, T], n)$. Let*

$$\phi(t, s, v) = \sum_{j=1}^{m} B(\phi_j(s, v), c_j(t)).$$

Let $\{z_1, \ldots, z_m\}$ be the solution to the following system of ODEs:

$$\frac{dz_j(t)}{dt} = \phi_j(t, u(t)) \text{ for all } t \in [t_0, T], \text{ with } z_j(t_0) = 0, \tag{3.10}$$

where

$$u(t) = \psi(t) + \sum_{j=1}^{m} B(z_j(t), c_j(t)).$$

Then, u, ϕ, ψ respect eq. (3.7).

Proof. Integrating eq. (3.10), we obtain

$$z_j(t) = \int_{t_0}^{t} \phi_j(s, u(s))ds.$$

Therefore:

$$\begin{aligned} u(t) &= \psi(t) + \sum_{j=1}^{m} B(z_j(t), c_j(t)) \\ &= \psi(t) + \sum_{j=1}^{m} \int_{t_0}^{t} B(\phi_j(s, u(s)), c_j(t))ds \\ &= \psi(t) + \int_{t_0}^{t} \sum_{j=1}^{m} B(\phi_j(s, u(s)), c_j(t))ds \\ &= \psi(t) + \int_{t_0}^{t} \phi(t, s, u(s))ds. \end{aligned}$$

□

This can be seen as a continuous analog of neural architecture search. Given a family of Neural ODEs $\{\phi_1, \ldots, \phi_m\}$, and functions $\{c_1, \ldots, c_m\}$, we can compute a loss function with respect to the Volterra machine

$$\sum_{j=1}^{m} B(\phi_j(s, u(s)), c_j(t)).$$

From this perspective, the relative strengths of $c_j(t)$ can be interpreted as *routing*. We will give an application of Volterra machines in section 3.2.3, in the context of kernel methods.

3.2.3 Kernel machines

We are interested in combining kernel methods [111] with the machine framework. In their simplest form, kernel methods associate to an input

space X a Hilbert space H of real-valued functions defined on X. Here, however, we are interested in studying Hilbert spaces of endofunctions of X. To do so, we will need some notions from the theory of *operator-valued* kernel methods [3, 64, 87].

Operator-valued kernels

Let X be a space, and Y a Hilbert space, with scalar product $\langle \text{-}, \text{-} \rangle$. We are interested in studying functions $X \to Y$. In the remainder, we will denote the set of functions from a space X to another space Y by Y^X. Let $\mathcal{L}(Y)$ be the space of bounded linear endomorphisms of Y. It is a Banach space, with norm given by the operator norm.

Definition 8. *[64, Def. 3] Let Y be a Hilbert space. A map $K \colon X \times X \to \mathcal{L}(Y)$ is an* operator-valued kernel *if the following conditions are satisfied.*

1. *For all $x_1, x_2 \in X$ the operator $K(x_1, x_2) \colon Y \to Y$ is self-adjoint.*
2. *For all $x_1, \ldots, x_n \in X$, $c_1, \ldots, c_n \in Y$, the matrix*

$$M_{i,j} = \langle c_i, K(x_i, x_j) c_j \rangle$$

   is positive-semidefinite.

Remark 2. *A scalar kernel on X can always be seen as an operator-valued kernel $K \colon X \times X \to \mathcal{L}(Y)$, where for all $x_1, x_2 \in X$, $K(x_1, x_2)$ is a multiple of the identity.*

An operator-valued kernel $K \colon X \times X \to \mathcal{L}(Y)$ will induce a feature map $X \to H \subseteq Y^X$, where H is the *Reproducing Kernel Hilbert Space* (RKHS [7]) associated to K. In particular, H is a space of Y-valued functions on X. Every function in H can be written as a sum:

$$f(x) = \sum_{j=1}^{\infty} K(x, x_j) c_j,$$

where, for every j, $x_j \in X$ and $c_j \in Y$. Even though the above sum has infinite elements, this is never a problem in practice. Given a function $f \in H$ and a finite dataset $\{x_1, \ldots, x_m\}$, one can always find $\{c_1, \ldots, c_m\}$ such that, for all $x \in \{x_1, \ldots, x_m\}$,

$$f(x) = \sum_{j=1}^{m} K(x, x_j) c_j.$$

In general machine learning problems, the function $\sum_{j=1}^{m} K(\text{-}, x_j) c_j$ is preferable to f as, even though they are indistinguishable on the training dataset, we have

$$\left\| \sum_{j=1}^{m} K(\text{-}, x_j) c_j \right\|_H \leq \|f\|_H ,$$

and hence $\sum_{j=1}^{m} K(\text{-}, x_j) c_j$ tends to be smoother and better behaved.

As H is a space of functions from X to Y, we have a canonical map $H \times X \to Y$, given by function evaluation. In what follows, we will focus on the case $X = Y$.

Definition 9. *Let X be a Hilbert space. Let $K \colon X \times X \to \mathcal{L}(X)$ be an operator-valued kernel, with RHKS H. K is a* kernel machine *if the canonical map*

$$H \times X \to X$$

is a parametric machine.

Definition 9 implies that for all $f \in H$, the function f is a machine on X. Furthermore, one can use standard techniques from kernel methods to learn a function $f \in H$ whose associated stable state optimizes some relevant quantity. In the case of kernel machines, an analog of the representer theorem [67] holds.

Theorem 4. *Let us consider a finite set $S = \{s_1, \ldots, s_m\}$, a map $g \colon S \to X$, and a function $\Lambda \colon X^m \times \mathbb{R} \to \mathbb{R}$ strictly increasing in the last variable.*

Any solution to the optimization problem

$$\min_{f \in H} \Lambda(h(s_1), \ldots, h(s_m), \|f\|_H), \tag{3.11}$$

where h is the stable state of f with initial condition g, is of the form

$$f(x) = \sum_{j=1}^{m} K(x, h(s_j))c_j.$$

Proof. Let us consider one solution f. Let h be its stable state with initial condition g, and let $x_j = h(s_j)$, for $j \in \{1, \ldots, m\}$. Let $\tilde{f}$ be the projection of f on the subspace:

$$\{K(\text{-}, x_1)c_1 + \cdots + K(\text{-}, x_m)c_m \,|\, c_1, \ldots, c_m \in X\}.$$

We start by observing that, for each $j \in \{1, \ldots, m\}$, $\tilde{f}(x_j) = f(x_j)$. h is the stable state of $\tilde{f}$ with initial condition g, as, for every $j \in \{1, \ldots, m\}$,

$$h(s_j) = g(s_j) + f(h(s_j)) = g(s_j) + \tilde{f}(h(s_j)).$$

As a consequence, $\tilde{f}$ produces a value smaller or equal than f in eq. (3.11), with equality if and only if they have the same norm, that is to say

$$f \in \{K(\text{-}, x_1)c_1 + \cdots + K(\text{-}, x_m)c_m \,|\, c_1, \ldots, c_m \in X\}.$$

□

In the context of kernel machines and for very small datasets, theorem 4 can be applied directly, guaranteeing optimality. In practice, for medium or large datasets, standard downsampling techniques, such as Nyström sampling [40], could be applied to replace $S = \{s_1, \ldots, s_m\}$ with a smaller subset of anchor points $\tilde{S} = \{\tilde{s}_1, \ldots, \tilde{s}_{\tilde{m}}\}$, with $\tilde{m} < m$.

In the following section 3.2.3, we will give two classes of examples of kernel machines, based on *discrete* and *continuous* filtrations of a Hilbert space.

Finite depth kernel machines

We associate a kernel machine to an arbitrary Hilbert space equipped with a finite filtration of closed subspaces.

Definition 10. *Let X be a Hilbert space, equipped with a finite filtration of closed subspaces*

$$0 = X_0 \subseteq X_1 \subseteq X_2 \cdots \subseteq X_n \subseteq X_{n+1} = X.$$

Let us consider a family of operator-valued kernels

$$K_i \colon X_i \times X_i \to \mathcal{L}(X_{i+1} \cap X_i^{\perp}) \text{ for } i \in \{0, \ldots, n\}.$$

The sum kernel machine *is given by*

$$K = \sum_{i=0}^{n} K_i.$$

The decomposition $K = \sum_{i=0}^{n} K_i$ corresponds to a decomposition of the RKHS $H \simeq \bigoplus_{i=0}^{n} H_i$, where, for every i, H_i is the RHKS of K_i. In particular, given an endofunction $f \in H$, we have a unique decomposition $f = f_0 + \cdots + f_n$, where $f_i \in H_i$ for all $i \in \{0, \ldots, n\}$.

Proposition 1. *Let K be a sum kernel machine, and let H be the corresponding RKHS. The application map*

$$\begin{aligned} \varrho \colon H \times X &\to X \\ (f, x) &\mapsto f(x) \end{aligned}$$

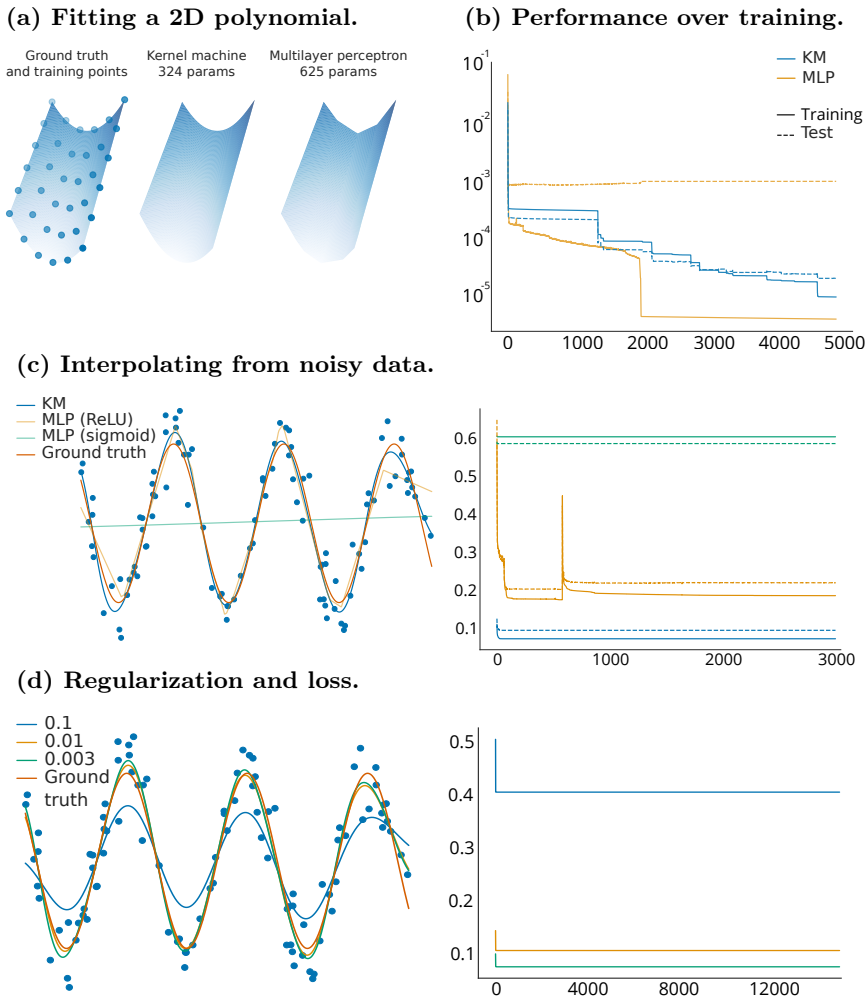


Figure 3.3: **Performance of finite-depth kernel machines.** We trained a kernel network and a multilayer perceptron with the same number of trainable parameters to fit a polynomial in two variables on a 6×6 grid of points. While both achieve good performance, the kernel machine shows better decoding a and a smaller loss function on the validation set (dashed line) b. In c we test robustness to noise of the kernel machine (514 parameters) in a noisy interpolation problem, comparing it with a 2 layers perceptron (609 parameters) with ReLU and sigmoid nonlinearities, respectively. In d we show how different regularization coefficients affect the performance of the kernel machine.

is a parametric machine. As a consequence, each endofunction $f \in H$ is a machine.

Proof. Let us write:

$$\varrho = \varrho_0 + \cdots + \varrho_n,$$

where, for $i \in \{0, \ldots, n\}$, ϱ_i is the application map corresponding to K_i. It is straightforward to show that, for $i_1 \leq i_2$, ϱ_{i_1} does not depend on ϱ_{i_2}. In particular, each ϱ_i is square-zero, and thus a machine. Moreover, the dependency graph of $\{\varrho_0, \ldots, \varrho_n\}$ is acyclic, as the source of each edge always has a smaller index than the target. It follows from corollary 1 that $\varrho_0 + \cdots + \varrho_n$ is a machine, whose stable state can be computed via eq. (3.6). □

In classical terms, kernel machines in H correspond to a network with $n + 1$ layers and *all* shortcut connections. While in classical deep neural networks this would cause an explosion in the number of parameters, which would grow quadratically with the number of layers, in the case of small datasets and kernel machines this is not the case. A general kernel machine, on a training set with m datapoints $\{s_1, \ldots, s_m\}$, can be expressed as:

$$X \ni x \mapsto \sum_{i=0}^{n} \sum_{j=1}^{m} K_i(x, h(s_j))c_j, \tag{3.12}$$

where h is the stable state of the kernel machine (see theorem 4). Each c_j is a vector of $\dim(X)$ free parameters. Therefore the number of parameters, $m \cdot \dim(X)$, grows linearly, rather than quadratically, with the number of layers.

Finite-depth kernel machines on small datasets. Small datasets are the natural testbed for finite-depth kernel machines given the architecture described by eq. (3.12) and optimality guarantees obtained in theorem 4.

We implemented this architecture as a PyTorch module and chose to work with radial basis function kernels of the form

$$K(u, v) = \exp(-\|u - v\|^2).$$

We first test the architecture on a surface-fitting task, with ground truth $p(x, y) = (2x - 1)^2 + 2y + xy - 3$. The training set consists of 36 points obtained by evaluating p on a uniform 6×6 grid in $[0, 1]^2$. Test points are randomly chosen in the same domain (see fig. 3.3a). We report the performance of the kernel machine (324 parameters) in fig. 3.3b and compare it with a two-layers perceptron (625 parameters). Although both architectures are regularized, we can observe how the perceptron's performance is affected by overfitting, while the kernel machine reaches similar loss values on the training and test set. We then test the same kernel machine on the interpolation of noisy data, see fig. 3.3c. Again, we compare its performance against 2-layer perceptrons with ReLU and sigmoid activation functions, respectively. We train on 100 random points obtained by sampling from a noisy sine. The kernel machine reaches the best performance on both the training and the validation set. Finally, on the same task, we test in fig. 3.3d the robustness of the kernel machine to variation of the regularization cost.

Infinite depth kernel machines

To translate the discrete filtration kernel described in section 3.2.3 to the continuous case, we replace the discrete filtration with a continuous one. Let X be a Hilbert space, $t_0 < T \in \mathbb{R}$, and

$$0 = X_{t_0} \subseteq \cdots \subseteq X_t \subseteq \cdots \subseteq X_T = X \tag{3.13}$$

a filtration of closed subspaces of X. We need a technical assumption to proceed in the continuous case.

Definition 11. *Let X be a Hilbert space. Let $\{X_t\}_{t\in[t_0,T]}$ be a filtration on X, and let π_t denote the orthogonal projection on X_t, for $t \in [t_0, T]$. We say that $\{X_t\}_{t\in[t_0,T]}$ is* continuous *if, for all $x \in X$, the function*

$$\begin{aligned}[t_0, T] &\to X \\ t &\mapsto \pi_t(x)\end{aligned}$$

is continuous with respect to the norm on X.

Theorem 5. *Let X be a Hilbert space, with a continuous filtration $\{X_t\}_{t\in[t_0,T]}$, and corresponding orthogonal projections $\{\pi_t\}_{t\in[t_0,T]}$. Let $K\colon X \times X \to \mathcal{L}(X)$ be an operator-valued kernel, and H be its RKHS. Finally, let*

$$\varrho\colon H \times X \to X$$

be the application map. Let us assume that

- *the distance induced by K is bounded by a multiple of the norm-induced distance on X,*
- *for all $t \in [t_0, T]$, for all $x_1, x_2 \in X$,*

$$K(x_1, x_2)\pi_t = \pi_t K(x_1, x_2) = \pi_t K(\pi_t x_1, \pi_t x_2).$$

Then ϱ is a parametric machine, that is to say K is a kernel machine.

Proof. Let $\lambda > 0$ be such that, for all $x_1, x_2 \in X$,

$$\|K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)\|_{\mathcal{L}(X)} \leq \lambda^2 \|x_1 - x_2\|_X^2.$$

Let

$$H \ni \xi = \sum_{j=1}^{\infty} K(\text{-}, x_j) c_j.$$

Let m be such that

$$\left\| \xi - \hat{\xi} \right\| \leq \frac{1}{4\lambda}, \text{ where } \hat{\xi} := \sum_{j=1}^{m} K(\text{-}, x_j) c_j.$$

As the filtration $\{X_t\}_{t \in [t_0, T]}$ is continuous, for all $c \in X$, the map $t \mapsto \pi_t(c)$ is continuous and, therefore, uniformly continuous. In addition, π_t commutes with K by hypothesis, therefore we can choose N such that, given $t_i := t_0 + \frac{i}{N}(T - t_0)$, for $i \in \{1, \ldots, N\}$,

$$\left\| \left(\pi_{t_i} - \pi_{t_{i-1}} \right) \hat{\xi} \right\| = \left\| \sum_{j=1}^{m} K(\text{-}, x_j) \left(\pi_{t_i} - \pi_{t_{i-1}} \right) c_j \right\| \leq \frac{1}{4\lambda}.$$

Let $B\left(\xi, \frac{1}{4\lambda}\right)$ be an open ball of radius $\frac{1}{4\lambda}$ around ξ. For each $\tilde{\xi} \in B\left(\xi, \frac{1}{4\lambda}\right)$, and for all $i \in \{1, \ldots, N\}$,

$$\left\| \left(\pi_{t_i} - \pi_{t_{i-1}} \right) \tilde{\xi} \right\| \leq \frac{3}{4\lambda}.$$

For all $i \in \{1, \dots, N\}$, for all $\tilde{\xi} \in B\left(\xi, \frac{1}{4\lambda}\right)$, let $\tilde{\xi}_i = \left(\pi_{t_i} - \pi_{t_{i-1}}\right)\tilde{\xi}$. Then, for all $x_1, x_2, c \in X$,

$$\begin{aligned}
\langle \tilde{\xi}_i(x_1 - x_2), c\rangle_X^2 &= \langle \tilde{\xi}_i, K(\text{-}, x_1)c - K(\text{-}, x_2)c\rangle_H^2 \\
&\leq \|\tilde{\xi}_i\|_H^2 \cdot \|K(\text{-}, x_1)c - K(\text{-}, x_2)c\|_H^2 \\
&\leq \left(\frac{3}{4\lambda}\right)^2 \cdot \langle c, \left(K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)\right)c\rangle_X \\
&\leq \left(\frac{3}{4\lambda}\right)^2 \cdot \|c\|_X^2 \|K(x_1, x_1) + K(x_2, x_2) - 2K(x_1, x_2)\|_{\mathcal{L}(X)} \\
&\leq \left(\frac{3}{4}\right)^2 \|c\|_X^2 \|x_1 - x_2\|_X^2.
\end{aligned}$$

By choosing $c = \tilde{\xi}_i(x_1 - x_2)$, it follows that

$$\|\tilde{\xi}_i(x_1 - x_2)\| \leq \frac{3}{4}\|x_1 - x_2\|,$$

hence $\tilde{\xi}_i$ is a contraction. For $i_1 < i_2$, $\tilde{\xi}_{i_1}$ does not depend on $\tilde{\xi}_{i_2}$, therefore

$$\tilde{\xi} = \tilde{\xi}_1 + \cdots + \tilde{\xi}_N$$

is a machine, thanks to corollary 1. As the same computing procedure can be applied to to all $\tilde{\xi}$ in a neighborhood of ξ, we have shown that $\varrho\colon H \times X \to X$ is a continuous parametric machine, hence K is a kernel machine. □

Computing continuous kernel machines efficiently

Let us consider a particular case of filtration on a Hilbert space. Let $X = L^2([t_0, T], n)$, and for all $t \in [t_0, T]$ let $X_t = L^2([t_0, t], n)$. Let $k\colon \mathbb{R}^n \times \mathbb{R}^n \to \mathcal{L}(\mathbb{R}^n)$ be a finite dimensional continuous operator-valued kernel. We can

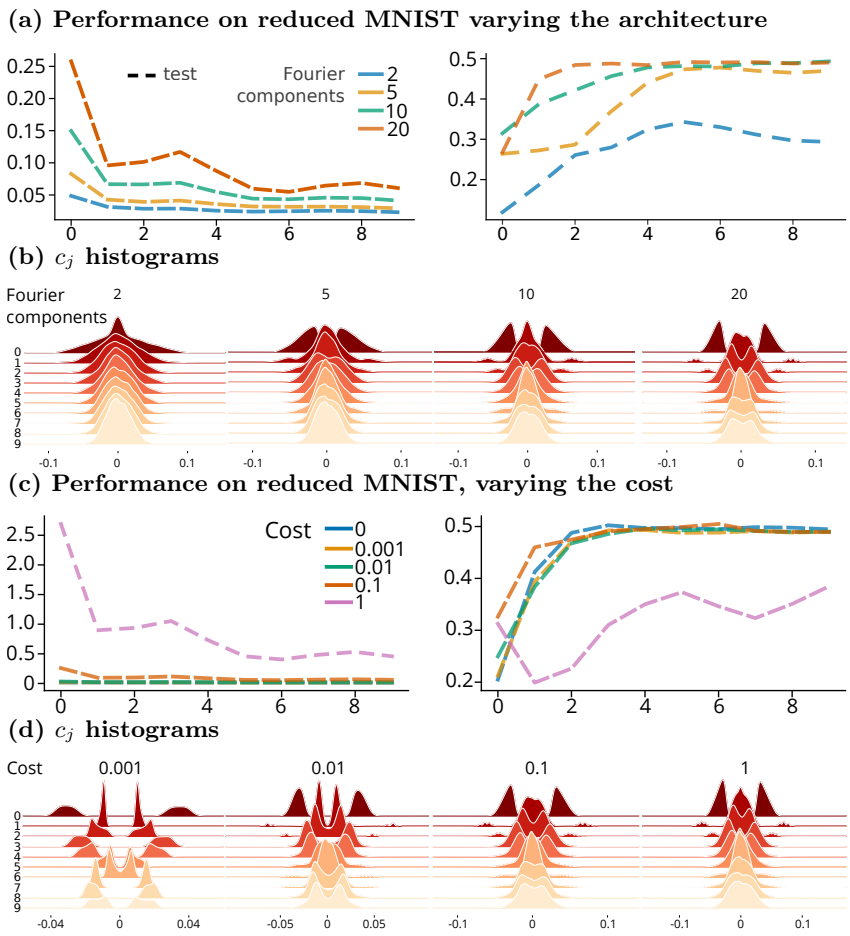


Figure 3.4: **Performance of infinite-depth kernel machines.** We trained an infinite-depth kernel machine on one random sample per class of the MNIST dataset and tested it on the 10000 test samples. We approximated c_j with a truncated Fourier series. In fig. 3.4a we report the test loss and accuracy obtained varying the number of Fourier components used to approximate c_j and fixing the regularization cost to 0.1. According to our expectation an increase in number of Fourier components corresponds to an increase in performance (best $\approx 50\%$). Figure 3.4b shows how c_j evolves during training (epochs on the y-axis). Setting the number of Fourier components to 20, the change in performance caused by a variation of the regularization cost is reported in fig. 3.4c, and in fig. 3.4d the corresponding c_j histograms.

consider the following operator-valued kernel

$$(K(x_1, x_2)x)(t) = \int_{t_0}^{t} k(x_1(s), x_2(s))x(t)ds,$$

where $x_1, x_2, x \in X$ and $t \in [t_0, T]$. Let H be the RKHS corresponding to K. Let us further assume that the distance induced by k on $\mathbb{R}^n$ is bounded by a multiple of the Euclidean distance. Then the distance induced by K on X is bounded by a multiple of the L^2 distance. By theorem 5, $H \times X \to X$ is a parametric machine.

Let $f = \sum_{j=1}^{m} K(\text{-}, x_j)c_j$ be a machine in H. Let $\psi \in L^2([t_0, T], n)$ be an initial condition. The stable state is given by the solution to the following Volterra equation:

$$u(t) = \psi(t) + \int_{t_0}^{t} \sum_{j=1}^{m} k(u(s), x_j(s))c_j(t)ds, \text{ for all } t \in [t_0, T], \qquad (3.14)$$

which can be computed efficiently using theorem 3.

Infinite-depth kernel machines on small datasets. Equation (3.14) gives an efficient way to implement and compute infinite-depth kernel machines. This construction satisfies the optimality result obtained in theorem 4. Thus, we implemented infinite-depth kernel machines as a PyTorch module and tested them on small datasets. Intuitively, an infinite-depth kernel machine is a continuous architecture adding to state-of-the-art implementation, such as neural ODEs, all shortcut connections in time. Thus, we test infinite-depth kernel machines on a reduced version of the MNIST dataset [39] obtained considering one random sample per class (i.e. ten training images). In the function space defined by the machine, the architecture is chosen by selecting an incomplete basis on which the parameters of the machine are expressed. In our simulations, we consider radial basis kernel functions and an incomplete Fourier basis. In fig. 3.4a, we report the

performance (loss and accuracy on the 10000 image MNIST test set) of the kernel machine when varying the architecture, i.e. varying the number of considered Fourier components. As expected, an increase in the number of such components causes an increase in performance. An histogram of the parameters c_j (see eq. (3.14)) is shown in fig. 3.4b. Figures 3.4c and 3.4d show the change in performance of the kernel machine while varying the regularization cost.

3.3 Discussion

We provide a solid categorical foundation for the study of deep neural networks. Borrowing ideas from functional analysis and category theory, we define the abstract notion of *machine*, whose *stable state* generalizes the computation of a feedforward neural network. It is a unified concept that encompasses both manually designed neural network architectures, as well as their continuous counterpart such as Neural ODEs [32].

We take as starting point a forgetful functor from a linear category (Banach spaces) to a nonlinear one (topological spaces). This alternation between linear and nonlinear components is one of the key ingredients of the success of deep neural networks, as it allows one to obtain complex functions as a composition of simpler ones. The notion of composition of layers in neural networks is unfortunately somewhat ill-defined, especially in the presence of shortcut connections and non-standard architectures. In the proposed *machine framework*, the composition is replaced by the sum. We describe independence conditions to ensure that the sum of machines is again a machine, in which case we can compute its stable state (forward pass) explicitly. This may seem counterintuitive, as the sum is a commutative operation, whereas the composition is not. However, in our framework, it is the *dependency graph* of a collection of machines that determines the order of composition.

Basic combinations of simple machines—*square-zero* and *contracting*—cover a lot of ground. In particular, using finite sums of square-zero machines (discrete architectures), we recover classical neural networks, including architectures with shortcut connections. In this setting, we provide a first simple application. Starting with a convolutional network with maximal connectivity, the architecture is automatically pruned during training until a minimal architecture with robust performance is found. This algorithm is available as a PyTorch module. Contracting, *infinite-depth* architectures generalize neural ODEs [32]. More generally, we prove that, under some Lipschitz and continuity conditions, nonlinear integral Volterra equations of the second kind are machines. We provide an efficient procedure, with corresponding PyTorch implementation, to solve such equations in a special case.

Our approach meshes well with *deep kernel learning* [33, 74, 83, 84, 91], an attempt to combine modern advances in deep learning with classical kernel methods [58]. We believe this is particularly promising when working with small datasets, a scenario where deep neural networks have traditionally been less successful. We introduce the notion of *kernel machine*, a Hilbert space whose points are machines. There, given a specific loss function, we can search for machines that minimize it and that have a small norm. Even though the space is potentially infinite-dimensional, we prove an analog of the representer theorem, which determines a finite-dimensional subspace where optimal solutions can be found. This subspace can be quite large in practice. However, the norm can be used to regularize solutions.

We propose and implement in PyTorch two examples of kernel machines, with finite- and infinite-depth. First, using kernels on finite filtrations of Hilbert spaces, we build finite-depth kernel machines. They correspond to neural networks with all shortcut connections. In our simulations, with a comparable number of trainable parameters, kernel ma-

chines outperform multilayer perceptrons in toy problems with no more than 100 training datapoints. Second, using continuous filtrations on function spaces, we build infinite-depth kernel machines. While preserving the advantages of a kernel-based approach (optimality guarantees), infinite-depth kernel machines introduce the concept of shortcut connection in neural ODEs. Indeed, given a kernel, the value of the stable state (output) of the machine at time t is obtained considering a restriction of the kernel to the interval $[t_0, t]$, i.e. all shortcuts up to time t.

The parameters to be optimized are functions in a Hilbert space. As mentioned above, the infinite-dimensional function space represents a continuous architecture with *all* shortcut connections. As a consequence, a key ingredient of this method is the choice of the incomplete (finite) basis used to approximate such functions, which corresponds to a choice of architecture. For instance, discrete architectures can be recovered in this infinite-depth framework, via an incomplete basis of piecewise constant functions (grid-based approximation). Rather than limiting the *depth* of our architecture by approximating its parameters on a grid, we choose a low-frequency approximation, working with truncated Fourier series. This is, to the best of our knowledge, a novel approach to *Neural Architecture Search* [43], where different architectures can be chosen (and compared) simply by selecting an incomplete basis of an infinite-dimensional function space.

3.4 Materials and methods

The simulations in figs. 3.2 to 3.4 are based on custom PyTorch [96] code, which has not yet been publicly released and is available upon request.

3.5 Author contributions

P.V., P.F and M.G.B devised the project. P.V. and M.G.B developed the mathematical framework. P.V. and M.G.B. developed the software to implement the framework. P.V. wrote the original draft. M.G.B. reviewed and edited.

Chapter 4

Learning to represent signals spike by spike

Summary

Networks based on coordinated spike coding can encode information with high efficiency in the spike trains of individual neurons. These networks exhibit single-neuron variability and tuning curves as typically observed in cortex, but paradoxically coincide with a precise, non-redundant spike-based population code. However, it has remained unclear whether the specific synaptic connectivities required in these networks can be learnt with local learning rules. Here, we show how to learn the required architecture. Use coding efficiency as an objective, we derive spike-timing-dependent learning rules for a recurrent neural network, and we provide exact solutions for the networks' convergence to an optimal state. As a result, we deduce an entire network from its input distribution and a firing cost. After learning, basic biophysical quantities such as voltages, firing thresholds, excitation, inhibition, or spikes acquire precise functional interpretations.

4.1 Introduction

Many neural systems encode information by distributing it across the activities of large populations of spiking neurons. A lot of work has provided pivotal insights into the nature of the resulting population codes [49, 119, 10, 147], and their generation through the internal dynamics of neural networks [4, 15, 41, 26]. However, it has been much harder to understand how such population codes can emerge in spiking neural networks through learning of synaptic connectivities [56].

For sensory systems, the efficient coding hypothesis has provided a useful guiding principle, which has been successfully applied to the problem of unsupervised learning in feedforward networks [14, 95]. When transferring the insights gained in these simplified rate networks to more realistic, biological networks, two key challenges have been encountered. The first challenge are locality constraints. Indeed, synapses have usually only access to pre- and postsynaptic information, but most unsupervised learning rules derived in rate networks use omniscient synapses that can pool information from across the network. In turn, the derivation of learning rules under locality constraints has often relied on heuristics or approximations [151, 110, 20, 27], although more recent work has shown progress in this area [139, 99, 100]. We note that supervised learning in neural networks faces similar problems, and recent work has sought to address these issues [142, 52, 107, 1, 73]. We will here focus on unsupervised learning.

The second challenge are spikes. Indeed, spikes have often proved quite a nuisance when moving insights from rate networks to spiking networks. In order to maintain the functionality of a given rate network, for instance, the equivalent spiking network usually sacrifices either efficiency or realism. In mean-field approaches, each rate unit is effectively replaced by tens or hundreds of (random) spiking neurons, so that the spiking network becomes a bloated and inefficient approximation of its rate counterpart

[102]. In the 'neural engineering framework', this excessive enlargement is avoided [41, 42]. However, the spike trains of individual neurons become quite regular, in contrast to the random, almost Poissonian statistics observed in most neural systems.

Some of these problems have recently been addressed in networks with tightly balanced excitation and inhibition [38, 19, 12, 30]. These networks can produce functionality with limited number of neurons and random spiking statistics. One of the key insights of this literature has been that each neuron's voltage should measure a part of the network's global objective, such as the efficiency of the emitted spike code.

However, it has largely remained unclear how networks of spiking neurons could move into this globally optimal regime, given that they are only equipped with local synaptic plasticity rules. We here show that the membrane voltage holds the key to learning the right connectivity under locality constraints. If we start with a randomly connected or unconnected neural network, and simply assume that each neuron's voltage represents part of the global objective, then the locally available quantities such as membrane voltages and excitatory and inhibitory inputs are sufficient to solve the learning problem. Using these ideas, we derive learning rules and prove their convergence to the optimal state. The resulting learning rules are Hebbian and anti-Hebbian spike-timing and voltage-dependent learning rules, and are guaranteed to generate highly efficient spike codes.

4.2 Results

We study a population of excitatory (E) neurons that are interconnected with inhibitory (I) interneurons (fig. 4.1a1). The excitatory neurons receive many input signals, $x_j(t)$, from other neurons within the brain, and we will ask how the neurons can learn to encode these signals efficiently in their spiking output. We will first develop a measure for the efficiency of

neural population codes, then show the connectivity structure of efficient networks, and then show how the respective connectivity can be learnt. In this work, we focus exclusively on the problem of encoding a set of signals, and we defer the problem of how to compute with signals to the discussion.

For concreteness, we will study networks of leaky integrate-and-fire neurons. Each neuron's membrane potential is driven by feedforward input signals, $x_j(t)$, which we will model as a leaky integral of input currents $c_j(t)$, and by recurrent inputs, that feed the output spike trains, $o_k(t)$, back into the network. For simplicity, we will ignore the inhibitory interneurons for now and treat them as simple relays (fig. 4.1a2). As a consequence, we allow the excitatory neurons to violate Dale's law, a problem we will come back to later. Formally, the membrane voltages of the excitatory neurons obey the equation

$$\frac{dV_i}{dt} = -V_i + \sum_{j=1}^{M} F_{ij} c_j(t) + \sum_{k=1}^{N} \Omega_{ik} o_k(t), \tag{4.1}$$

where F_{ij} are the feedforward weights, and Ω_{ik} contains the recurrent synapses (for $i \neq k$) and the voltage resets (for $i = k$). A spike is fired when the voltage surpasses a threshold, T_i. The voltage is then reset to the value $V_i = T_i + \Omega_{ii}$, and we assume that $\Omega_{ii} < 0$. For simplicity, here we consider *instantaneous* synaptic transmission: the impact of synaptic delays on the network will be examined in fig. 4.7.

The first objective of the network will be to encode the input signals into a spiking output such that a downstream observer can reconstruct the input signal through a linear readout, i.e., a weighted sum of the neural responses (fig. 4.1b). We define this linear readout as

$$\hat{x}_j(t) = \sum_{k=1}^{N} D_{jk} r_k(t)\,, \tag{4.2}$$

where $r_k(t)$ is the postsynaptically filtered spike train of the k-th excitatory neuron, and D_{jk} is the decoding weight associated with the j-th signal.

The second objective of the network will be to find, among all possible spiking outputs, and all possible decoders, the ones that are the most efficient. We define the coding efficiency of the population as a trade-off between the accuracy and the cost of the generated code,

$$E = \Big\langle \sum_{j=1}^{M} (x_j - \hat{x}_j)^2 + C(r) \Big\rangle, \tag{4.3}$$

where the angular brackets denote averaging over time. The first term measures the accuracy of the code, given by the mean-squared error between the input signals and the linear readout. The second term, $C(r)$, denotes the cost of the code, exemplified for instance by the number of spikes fired. The smaller the loss, the higher the coding efficiency (see Supplementary Text S1 for details).

4.2.1 Efficient spike coding requires balance of excitation and inhibition

To find the most efficient spiking output, our network will need to modify its synapses. Since a single synapse can only see its pre- and postsynaptic partners and their relative spike trains, it cannot perceive the coding efficiency of the whole network. Without that information, it is unclear how the synapse should modify its weights in order to improve the coding efficiency. This rift between locally available information and global objective is the key conundrum of synaptic plasticity.

However, imagine we could intervene and simply set each neuron's recurrent synaptic weights such that they become equal to the feedforward weights multiplied by the decoding weights of a downstream observer, i.e., $\Omega_{ik} = -\sum_j F_{ij} D_{jk}$. As shown in Supplementary Text S2 and S3, the

membrane potential of each neuron can then be rewritten as

$$V_i(t) = \sum_{j=1}^{N} F_{ij}\big(x_j(t) - \hat{x}_j(t)\big). \tag{4.4}$$

In other words, given this specific connectivity structure, each neuron's membrane potential suddenly reflects a component of the global coding error, given by the difference between the input signals, $x_j(t)$, and the linear readout of a hypothetical downstream area, $\hat{x}_j(t)$. This peculiar structure emerges even though the membrane potential is generated from only feedforward and recurrent inputs (fig. 4.1a2,a3). Since synaptic plasticity can sense postsynaptic voltages, synapses have gained unexpected access to a component of the global coding error.

Moreover, each neuron will now bound its component of the error from above. Each time the error component becomes too large, e.g., due to an excitatory signal input, the membrane potential reaches threshold, and the neuron fires. The spike changes the readout, and, the global coding error decreases (under reasonable conditions on F_{ij} and D_{jk}, see Supplementary Text S4). This decrease in error is then signaled throughout the network. First, the firing neuron resets its own voltage after the spike, thus signaling to itself that its error component has decreased. Second, the firing neuron inhibits (or excites) all neurons with similar (or opposite) feedforward inputs, thus signaling them the decrease in error. The concurrent change in their respective membrane voltages is proportional to the overlap in information and thereby reflects the required update of the error components they are responsible for.

As a consequence, excitatory inputs that depolarize the membrane potential signal growing coding errors. Vice versa, inhibitory inputs that repolarize the membrane potential signal shrinking coding errors. In turn, when coding errors are kept in check, each feedforward excitatory input will be counterbalanced by a recurrent inhibitory input (and vice versa).

This latter reasoning links the precision of each neuron's code to the known condition of excitatory and inhibitory (EI) balance [38, 138, 6, 116, 103]. Indeed, if excitatory and inhibitory inputs are balanced optimally, the variance of the membrane potential, and thus, a neuron's error component, is minimized.

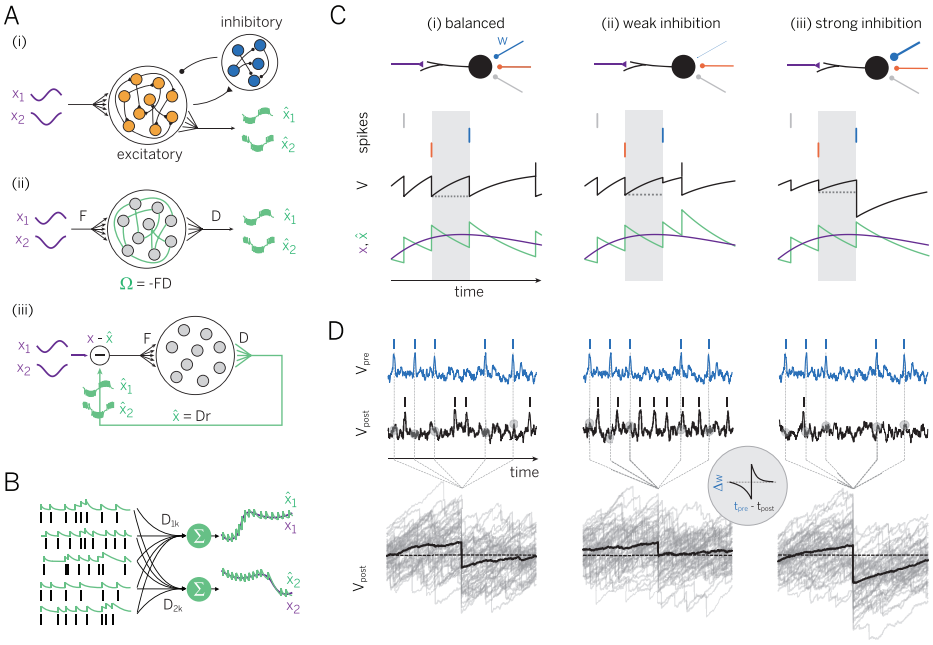


Figure 4.1: (legend on next page)

4.2.2 Recurrent synapses learn to balance a neuron's inputs

How can a network of neurons learn to move into this very specific regime? Several learning rules for EI balance have been successfully proposed before [123, 140], and spike-timing-dependent plasticity (STDP) can even balance

Figure 4.1: **Learning to represent analog signals efficiently with spikes.** (a1) Recurrent neural network with input x (purple) and signal estimates $\hat{x}$ (green), reconstructed from the spike trains of the excitatory population (see panel b). (a2) Simplified network without separate excitatory and inhibitory populations. (F=matrix of feedforward weights, D=matrix of decoding weights, Ω=matrix of recurrent weights). A network that has learnt to represent its input signals efficiently should have connectivity $\Omega = -FD$. (a3) Same as (a2), but unfolded to illustrate the effect of the recurrent connections. These connections act to subtract the reconstructed signal estimate from the incoming signal. As a consequence, the net input into each neuron is (a projection of) the reconstruction or coding error $x - \hat{x}$. (b) Linear readout of an analog signal from a population of spike trains. The spike train of each neuron is first filtered with a postsynaptic potential (left). The filtered spike trains are then linearly combined via decoding weights D to yield a signal estimate (right, green traces). (c) Learning of recurrent connections based on balancing the EI currents into each neuron spike by spike. Shown are the neuron's membrane voltage (black), which reflects the coding error, spikes from three inhibitory neurons (vertical lines, color-coded by connection), and the signal (purple), and signal estimate (green). (c1) Ideal case with EI balance. Each inhibitory spike perfectly counter-balances the prior excitatory drive. (c2) One inhibitory synapse too weak. The excitatory drive is not perfectly cancelled, resulting in an aberrant, early spike. (c3) One inhibitory synapse too strong. The excitatory drive is over-compensated, resulting in a prolonged hyperpolarization and a delay in subsequent spiking. (d) Learning of recurrent connections based on minimizing voltage fluctuations. Shown are the voltages and spikes of a pre- and a postsynaptic neuron over a longer time window (top) and the postsynaptic voltage fluctuations aligned at the timing of spikes from the presynaptic neuron (bottom, grey lines), as well as their average (bottom, black line). (d1) Ideal case with EI balance. Here, the average effect of the presynaptic spike is to turn a depolarized voltage into an equivalent hyperpolarized voltage (bottom panel, black line). (d2) If the inhibitory synapse is too weak, the average membrane voltage remains depolarized. (d3) If the inhibitory synapse is too strong, the average membrane voltage becomes overly hyperpolarized. (Inset: effect of the derived recurrent plasticity rule when tested with a paired-pulse protocol)

EI currents on a short time scale [140]. We will show that learning to balance excitatory and inhibitory inputs does indeed lead to the right type of connectivity (fig. 4.1a2-3), as long as EI currents are balanced as precisely as possible. Learning to balance avoids the pitfalls of a direct optimization of the coding efficiency with respect to the decoder weights, which is mathematically possible, but biophysically implausible for the synapses we consider here (see Supplementary Text S5). We developed two ways of reaching the balanced regime (see Supplementary Text S6 for a high-level, technical overview). The first scheme balances excitatory and inhibitory currents on a fine time scale (see Supplementary Text S7 and S8 for details), while the second scheme minimizes the voltage flucutations (see Supplementary Text S9–S12 for details). We here briefly explain the current-based scheme, but then focus on the voltage-based scheme for the rest of the text.

The first scheme directly targets the balance of excitatory and inhibitory currents. In fig. 4.1c, we show a neuron that receives excitatory feedforward inputs and inhibitory recurrent inputs. In the interval between two inhibitory spikes, the neuron integrates its excitatory feedforward input currents, which leads to a transfer of electric charges across the membrane (fig. 4.1c1, gray area). When the next inhibitory spike arrives (fig. 4.1c1, blue), electric charges are transfered in the opposite direction. Precise EI balance is given when these two charge transfers cancel exactly. When the second inhibitory spike overshoots (undershoots) its target, then the respective synaptic weight was too strong (weak), see fig. 4.1c2-3. To reach precise EI balance, this weight therefore needs to be weakened (strengthened). This learning scheme keeps the neuron's voltage (and thereby its component of the coding error) perfectly in check (see Supplementary Text S7 and S8 for details). We note that the membrane potential shown in fig. 4.1c is an illustrative toy example, for a network representing only one input signal with four neurons. In larger networks

that represent several input signals, the membrane potentials become more complex, and the inhibitory inputs due to recurrent connections become weaker than the voltage reset after a spike (see also below).

The precise accounting of charge balances across the membrane may seem unfeasible for real neurons. Our second scheme minimizes charge imbalances by confining deviations from a neuron's resting potential. If a recurrent weight is set such that each presynaptic spike, on average, resets a voltage depolarization to an equivalent hyperpolarization (or vice versa), then the membrane voltage is maximally confined (see fig. 4.1d). To move a recurrent synapse into this state, its weight should be updated each time a spike from presynaptic neuron k arrives, so that

$$\begin{aligned} \frac{d\Omega_{ik}}{dt} &\propto \text{pre} \times \text{post} \\ &= -o_k \left(2V_i + \Omega_{ik}\right). \end{aligned} \tag{4.5}$$

where o_k is the presynaptic spike train and V_i is the postsynaptic membrane potential before the arrival of the presynaptic spike. According to this rule, the recurrent connections are updated only at the time of a presynaptic spike, and its weights are increased and decreased depending on the resulting postsynaptic voltage. While this rule was derived from first principles, we note that its multiplication of presynaptic spikes and postsynaptic voltages is exactly what was proposed as a canonical plasticity rule for STDP from a biophysical perspective [34]. A minor difference to this biophysically realistic, 'bottom-up' rule, is that our rule treats LTP and LTD under a single umbrella. Furthermore, our rule does not impose a threshold on learning.

Once a synapse has been learnt with this voltage-based learning rule, it will tightly confine all voltage fluctuations as much as possible. This average confinement is illustrated in fig. 4.1d. We note that the membrane

potentials look more realistic here simply because the illustration is based on the simulation of a larger network with multiple input signals.

The learning rule drives the recurrent weights to the desired connectivity, given by the multiplication of the feedforward weights, F_{ij}, with an (a priori unknown) decoder matrix, D_{jk}, see fig. 4.1a2-3. To gain some intuition as to why that is the case, we will show that this connectivity structure is a stationary point of the learning rule. At this stationary point, the recurrent weights are no longer updated and become proportional to the average postsynaptic voltage of neuron i, $\Omega_{ik} = -2\langle V_i \rangle_k$, where the average, denoted by the angular brackets, is taken over all time points directly before the arrival of a spike from the presynaptic neuron k (see fig. 4.1d1). Since, whenever $\Omega_{ik} = -\sum_j F_{ij} D_{jk}$, the connectivity structure dictates that the voltage becomes a function of the global coding error, as stated in eq. (4.4), the stationary point can be rewritten as $\Omega_{ik} = -2\sum_j F_{ij}\langle x_j - \hat{x}_j \rangle_k$. If we now simply define the decoder matrix as $D_{jk} = 2\langle x_j - \hat{x}_j \rangle_k$, then $\Omega_{ik} = -\sum_j F_{ij} D_{jk}$. Accordingly, the peculiar multiplicative form of the recurrent weights, which transformed the voltage into a component of the coding error, is a stationary point of the learning rule (see Supplementary Text S9 for details and an additional convergence proof).

Depending on the precise cost terms, $C(r)$, required by the loss function, the learning rules undergo slight modifications. The effect of these cost terms is to penalize both the total number of spikes fired by the network, as well as high firing rates in individual cells. The learning rules used in all simulations are of the form

$$\frac{d\Omega_{ik}}{dt} \propto -o_k \Big(\beta(V_i + \mu r_i) + \Omega_{ik} + \mu \delta_{ik} \Big). \tag{4.6}$$

with β and μ positive constants, and with δ_{ik} the Kronecker delta (see Supplementary Text S9–S13 for a detailed explanation of these modifications and their relation to the cost).

Figure 4.2 illustrates the effect of the voltage-based learning rule in a network with 20 neurons receiving two random, uncorrelated feedforward inputs (see Supplementary Text S14 for details on the simulations). Since each neuron receives two input signals, each neuron has two feedforward weights. The initial setting of these weights was lopsided, as shown in fig. 4.2b1 (left panel), so that no neuron received a positive contribution of the first input signal. The recurrent weights were initially set equal to zero (fig. 4.2b1, right panel; the diagonal elements correspond to the self-resets of the neurons).

While the network receives the random input signals, the recurrent synapses change according to the learning rule, eq. (4.6), and each neuron thereby learns to balance its input currents. Once learnt, the recurrent connectivity reaches the desired structure, and the voltages of the neurons become proportional to a component of the coding error. As a result of the EI balance, the voltage fluctuations of individual neurons are much better bounded around the resting potential (compare fig. 4.2e1 with fig. 4.2e2), the global coding error decreases (fig. 4.2a), and the network experiences a large drop in the overall firing rates (fig. 4.2a,d1-2). The network's coding improvement is best illustrated in fig. 4.2c1-2, where we test the network with two input signals, a sine and cosine, and illustrate both the input signals and their reconstructions, as retrieved from the spike trains in fig. 4.2d1-2 using an optimal decoder. Note that this improvement occurred despite a drastic drop in overall firing rates (fig. 4.2d1-2).

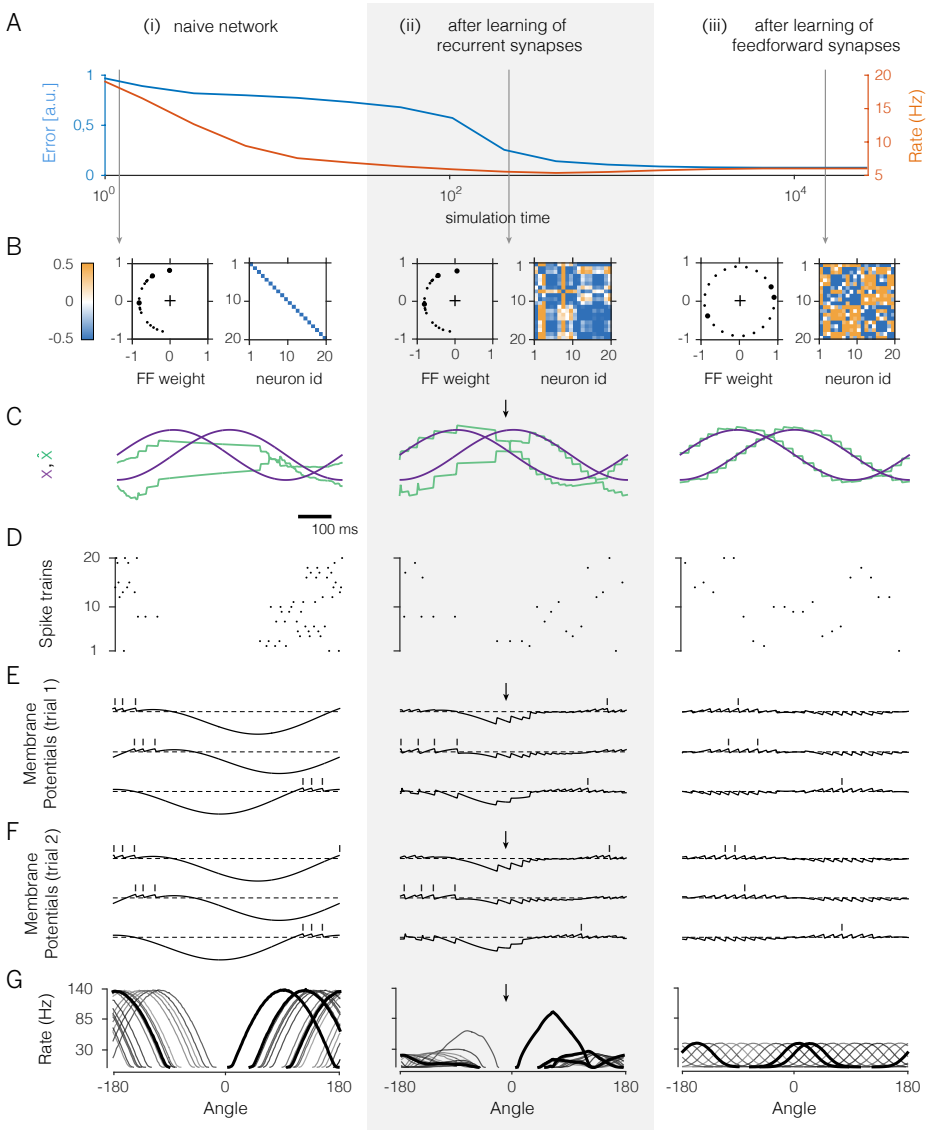


Figure 4.2: (legend on next page)

Figure 4.2: **A 20-neuron network that learns to encode two randomly varying signals.** (a) Evolution of coding error (blue), defined as the mean-square error between input signal and signal estimate, and mean population firing rate (orange) over learning. (b) Feedforward and recurrent connectivity at three stages of learning. In each column, the left panel shows the two feedforward weigths of each neuron as a dot in a two-dimensional space, and the right panel shows the matrix of recurrent weights. Here, off-diagonal elements correspond to synaptic weights (initially set to zero), and diagonal elements correspond to the neurons' self-resets after a spike (initially set to -0.5). (c) Time-varying test input signals (purple) and signal estimates (green). The test signals are a sine wave and a cosine wave. Signal estimates in the naive network are constructed using an optimal linear decoder. Arrows indicate parts of the signal space that remain poorly represented, even after learning of the recurrent weights. (d) Spike rasters from the network. (e) Voltages and spike times of three exemplary neurons (see thick dots in panel b). Dashed lines illustrate the resting potential. Over the course of learning, voltage traces become confined around the resting potential. (f) As in (e), but for a different trial. (g) Tuning curves (firing rates as a function of the angle of an input signal with constant radius in polar coordinates for all neurons in the network. Angles from $-90°$ to $90°$ correspond to positive values of x_1 which are initially not represented (panel b).

4.2.3 Feedforward weights change to strengthen postsynaptic firing

Despite the performance increase, however, the network still fails to represent part of the input, even after the recurrent connections have been learnt (fig. 4.2c2, arrow). Indeed, in the example provided, positive values of the first signal cannot be represented, because there are no neurons with positive feedforward weights for the first signal (fig. 4.2b1-2). These missing neurons can be easily spotted when plotting the tuning curves of all neurons (fig. 4.2g1-2). Here, directions of the input signal associated with positive values of the first signal are not properly covered, even after the recurrent weights have been learnt (fig. 4.2g2, arrow).

Consequently, the feedforward connections need to change as well, so that all parts of the input space are dealt with. We can again obtain a crucial insight by considering the final, 'learnt' state, in which case the feedforward connections are directly related to the optimal decoding weights. For example, if the input signals are mutually uncorrelated, i.e., $\langle x_i(t)x_j(t)\rangle = 0$ for zero-mean inputs and $i \neq j$, then the optimal feed-forward and decoding weights are equal, i.e., $F_{ik} = D_{ki}$ (see Supplementary Text S4). In section 4.2.3a, we illustrate the problem with five neurons that seek to represent two input signals. We assume a constant input signal, which we represent by a point in a signal space (section 4.2.3a1, purple dot). In turn, a neuron's spiking shifts the signal estimate in a direction given by its respective decoding weights, which we can illustrate through vectors (section 4.2.3a1, colored arrows). Accordingly, the input signal can be represented by a linear combination of the decoding vectors. For a biased distribution of decoding vectors, some input signals will require the combined effort of many neurons (section 4.2.3a1). For uncorrelated input signals, however, the best representation is achieved when the decoding vectors (and thereby the feedforward weights) are evenly distributed (section 4.2.3a2).

The feedforward weights of the i-th neuron can learn to optimally cover the input space if they change each time neuron i fires a spike,

$$\frac{dF_{ij}}{dt} \propto (x_j - \alpha F_{ij})\, o_i, \tag{4.7}$$

where x_j is the feed-forward input signal, α is a positive constant whose value depends on the enforced cost (see Supplementary Text S11), and o_i is the neuron's spike train. Note that the feedforward weights remain unchanged if neuron i does not spike.

The intuition for this rule is shown in section 4.2.3b. In an unconnected network, a neuron fires the most when its feedforward input drive is maximal. Under a power constraint on the input signal, the drive is maximized when the vector of input signals aligns with the vector of feedforward weights. In a network connected through recurrent inhibition, however, neurons start competing with each other, and a neuron's maximum firing (section 4.2.3b1; dashed lines) can shift away from the maximum input drive (section 4.2.3b1, colored arrows) towards stimuli that face less competition. If competition is well-balanced, on the other hand, then a neuron's maximum firing will align with the maximum input drive, despite the presence of recurrent connections (section 4.2.3b2, compare colored arrows and dashed lines). The above learning rule moves the network into this regime by shifting the feedforward weights towards input signals that elicit the most postsynaptic spikes (section 4.2.3b2, gray arrows). Learning converges when all tuning curve maxima are aligned with the respective feedforward weights (section 4.2.3b2; dashed lines and arrows). Eventually, the input space is thereby evenly covered (see Supplementary Text S10 for mathematical details).

From the perspective of standard frequency-modulated plasticity, the learning rule is Hebbian: whenever neuron i fires a spike, the resulting change in its synaptic weight F_{ij} is proportional to the j-th presynaptic

input, x_j, received at that time. The more neuron i spikes, and the higher the input x_j, the stronger the change in weight. Accordingly, connections are reinforced for co-occurring high pre- and postsynaptic activity. In the case of correlated input signals, the term "F_{ij}" is replaced by the covariance of the j-th presynaptic input signal with the total postsynaptic input current (see Supplementary Text S12). In this case, the decoding weights provide optimal coverage by favoring more frequent input signal directions (see section 4.2.3c,d).

The effect of the feedforward plasticity rule is shown in fig. 4.2a3–g3. The feedforward weights change slowly until the input space is spanned more uniformly (fig. 4.2b3). While these changes are occurring, the recurrent weights remain plastic on a faster time scale and thereby keep the system in a balanced state. At the end of learning, the neuron's tuning curves are uniformly distributed (fig. 4.2g3), and the quality of the representation becomes optimal for all input signals (fig. 4.2a3,c3). More specifically, the feedforward weights have become identical to the decoding weights, $F_{ik} = D_{ki}$, and the latter minimize the objective function, eq. (4.3).

Importantly, the final population code represents the input signals spike by spike, with a precision that approaches the discretization limit imposed by the spikes, i.e., the unavoidable steps in the signal estimate caused by the firing of individual spikes. Initially, when the neurons were unconnected (fig. 4.2b1), their voltages reflected the smooth, time-varying input (fig. 4.2e1). Moreover, neurons fired their spikes at roughly the same time from trial to trial (compare fig. 4.2e1 with fig. 4.2e2). After learning, the membrane potentials are correlated, reflecting their shared inputs, yet the individual spikes are far more susceptible to random fluctuations (compare fig. 4.2e3 with fig. 4.2f3). Indeed, whichever neuron happens to fire first immediately inhibits (resets) the others, so that a small initial difference in the membrane potentials is sufficient to change the firing order com-

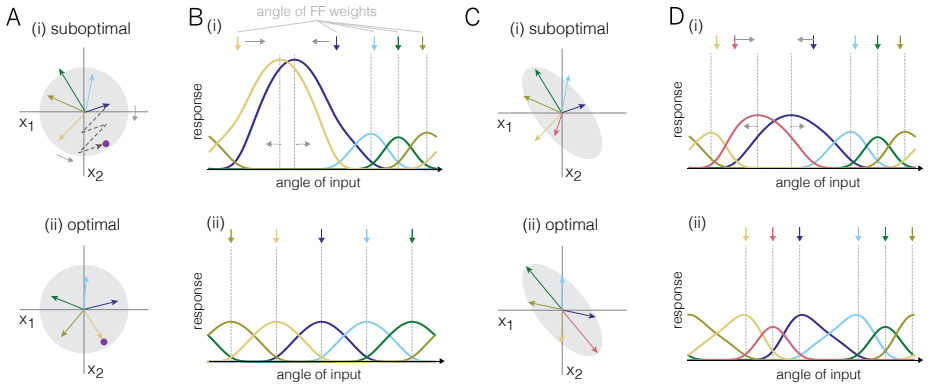


Figure 4.3: **Learning rules for the feedforward weights, depicted for a network.** (a) Arrangement of decoding weights influences coding efficiency. The purple dot represents the input signals, and each vector represents the jump in the signal estimates caused by the firing of one neuron. The gray circle represents the distribution of input signals; here, they are centered and uncorrelated. (a1) A biased arrangement of the decoding weights is suboptimal for uncorrelated signals. Many spikes are required to represent the purple input. (a2) Evenly spaced decoding weights are optimal for uncorrelated signals. Here, the purple input can be reached with a single spike. (b) Tuning curves of the five neurons before and after training. Shown are the firing rates of the neurons as a function of the angle of the input signal. Colored arrows above represent the feedforward weights (or the input signals that drive the neurons maximally in the absence of recurrent connections). (b1) In the untrained network, maximum input drive and maximum firing are not aligned. The learning rule shifts the feedforward weights towards the maximum of the firing rates (gray arrows, top). In turn, the firing rate maxima shift in the opposite direction (gray arrows, bottom). (b2) After learning, the maximum input drive (and thereby the feedforward weights) are aligned with the maximum firing rate. (c) Similar to (a), but for correlated input signals. (d) Similar to (c), but for correlated input signals. In the optimal scenario, the neurons' feedforward and decoding weights are attracted towards more frequent stimuli.

pletely. Here, the random nature of spike timing is simply a consequence of a mechanism that prevents any redundant (or synchronous) spikes. More generally, any source of noise or dependency on previous spike history will

change the firing order, but without a significant impact on the precision of the code. Thus, variable spike trains co-exist with a highly reproducible and precise population code.

4.2.4 Networks with separate excitatory and inhibitory populations

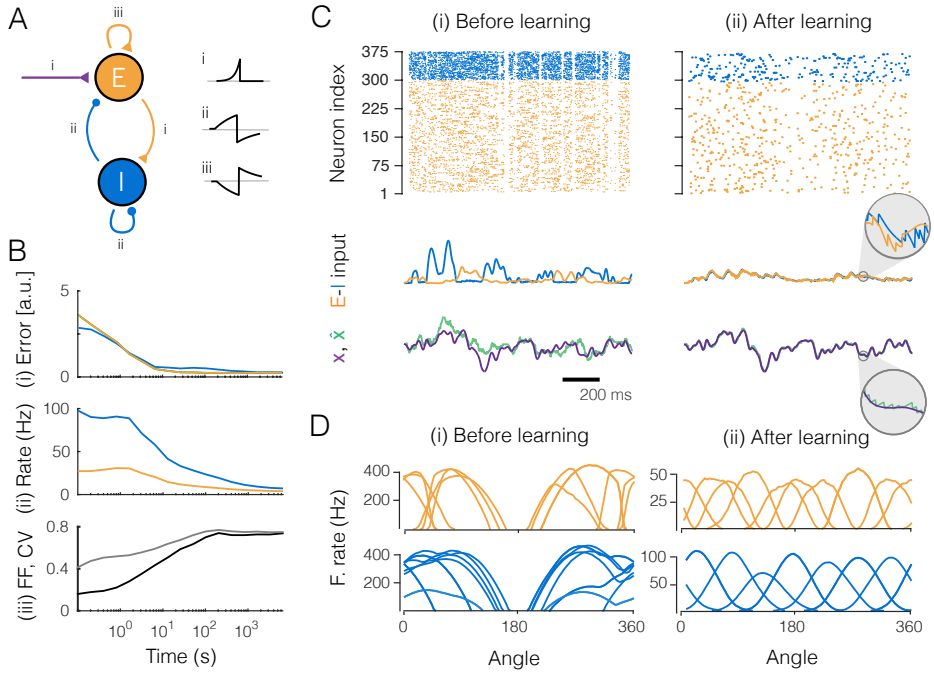


Figure 4.4: (legend on next page)

We have so far ignored Dale's law so that individual neurons could both excite and inhibit other neurons. Fortunately, all of our results so far can also be obtained in networks with separate excitatory (E) and inhibitory (I) populations (fig. 4.1a1), governed by eq. (4.1). In this more realistic case, the inhibitory population must simply learn to represent

Figure 4.4: **Large network (300 excitatory and 75 inhibitory neurons).** (a) The EI network as in fig. 4.1a1 and the learning rules (a1, feedforward rule; a2 and a3, recurrent rule). The insets show cartoon illustrations of the learning rules, stemming from STDP-like protocols between pairs of neurons, with the x-axis representing the relative timing between pre- and postsynaptic spikes ($\Delta t = t_{\text{pre}} - t_{\text{post}}$), and the y-axis the change in (absolute) weight. Note that increases (decreases) in synaptic weights in the learning rules map onto increases (decreases) for excitatory weights and decreases (increases) for inhibitory weights. This sign flip explains why the STDP-like protocol for EE connections yields a mirrored curve. (b) Evolution of the network during learning. (b1) Coding error for excitatory and inhibitory populations. The coding error is here computed as the mean square error between the input signals and the signal estimates, as reconstructed from the spike trains of either the excitatory or the inhibitory population. (b2) Mean firing rate of excitatory and inhibitory populations. (b3) Averaged coefficient of variation (CV, gray) and Fano factor (FF, black) of the spike trains. (c) Network input and output before (c1) and after (c2) learning. (Top) Raster plots of spike trains from excitatory and inhibitory populations. (Center) Excitatory and inhibitory currents into one example neuron. After learning, inhibitory currents tightly balance excitatory currents (inset). (Bottom) One of the three input signals (purple) and the corresponding signal estimate (green) from the excitatory population. (d) Tuning curves (firing rates as a function of the angle for two of the input signals, with the third signal clamped to zero) of the most active excitatory and inhibitory neurons.

the population response of the excitatory population, after which it can balance the excitatory population in turn. This can be achieved if we train the EI connections using the feedforward rule (eq. (4.7)) while the II, EE, and IE connections are trained using the recurrent rule (eq. (4.6); see Supplementary Text S13 for details).

Figure 4.4 illustrates how the key results obtained in fig. 4.2 hold in the full EI network. The network converges to the optimal balanced state (fig. 4.4b), and the precision of the representation improves substantially and approaches the discretization limit (fig. 4.4b1, c2), despite the overall decrease in output firing rates (fig. 4.4b2, c2). Initially regular and reproducible spike trains (fig. 4.4b3) become asynchronous, irregular, and comparable to independent Poisson processes (fig. 4.4b3, pairwise correlations are smaller than 0.001). Crucially, both the inhibitory and excitatory populations provide an accurate representation of their respective input signals, as shown by their small coding errors (fig. 4.4b1). Furthermore, we observe that the neurons' tuning curves, when measured along the first two signal directions, are bell-shaped just as in the previous example (fig. 4.4d2). Note that the inhibitory neurons fire more and have broader tuning than the excitatory neurons. This result is simply owed to their smaller number: since less neurons are available to span the signal space with their feedforward weights, they generally face less competition, and consequently have broader tuning.

4.2.5 Learning for correlated inputs

We have so far considered input signals that are mutually uncorrelated. For correlated input signals, the network learns to align its feedforward weights to the more frequent signal directions (section 4.2.3c). As a result, the tuning curves of the learnt network reflect the distribution of inputs experienced by the network (section 4.2.3d). In particular, tuning curves

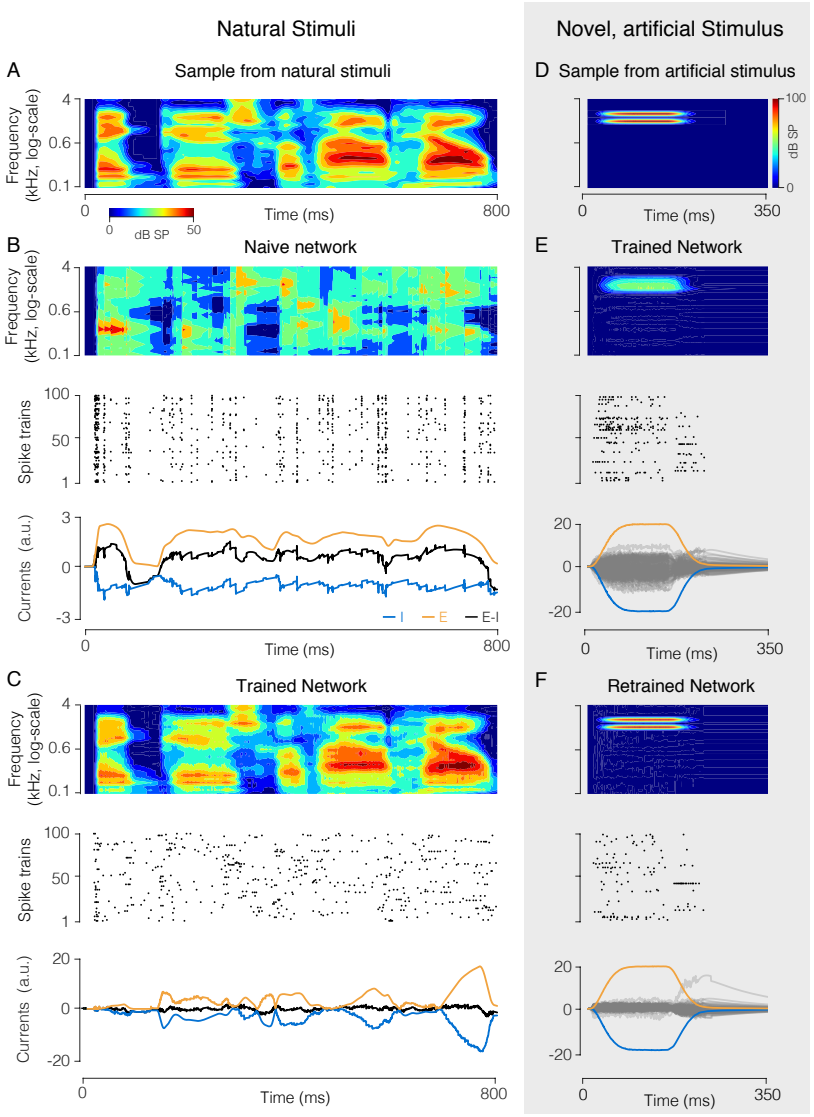


Figure 4.5: (legend on next page)

Figure 4.5: **Network (100 neurons) that encodes a high-dimensional, structured natural input (speech sounds).** (a) Spectrogram of a speech sound. (b) "Naive" network with random feedforward and recurrent weights. (Top) Optimal linear estimator applied to output spike trains reconstructs the stimulus poorly. (Center) Spike raster from all neurons, showing synchronous firing. (Bottom panel) Excitatory (orange) and inhibitory (blue) current into an example neuron are poorly balanced, causing large fluctuations in the total current (black). (c) Same as (b), after learning. The signal estimate tracks the signal closely (top), spike trains are asynchronous and irregular (center), and EI currents are tightly balanced (bottom). (d) Spectrogram of artificial, "non-speech" sound. (e) Response of the trained network trained to a non-speech sound, similar format as (b), (c). The new sound is improperly reconstructed (top), and EI currents of individual neurons are poorly balanced (bottom). Grey lines show the total (E+I) currents for the individual neurons, orange and blue lines show the mean excitatory and inhibitory currents, averaged over the population. (f) Same as (e), after re-training the network with a mixture of speech sounds and the new sound. The new sound is now represented precisely (top) with fewer spikes (center), and EI balance is improved (bottom).

are denser and sharper for signal directions that are a-priori more probable. This result is reminiscent of the predictions for efficient rate-based population codes with independent Poisson noise [48]. Note, however, that our networks learn a spike-per-spike code far more precise and efficient than such rate-based population codes.

To further demonstrate the power of the learning rules, using learning rules developed in Supplementary Text S12, we trained a network to represent speech signals, filtered through 25 frequency channels, in its spiking output (fig. 4.5a). Despite consisting of 100 neurons that fire at only ~ 4 Hz, the network learns to represent the signals with high precision (fig. 4.5b,c). This feat would be impossible if the network had not learnt the strong correlations in speech. After training with the speech signals, the feedforward and decoding weights adopt a structure reflecting the natural statistics of speech. The feedforward weights typically have excitatory subfields covering a limited range of frequencies, as well as inhibitory sub-

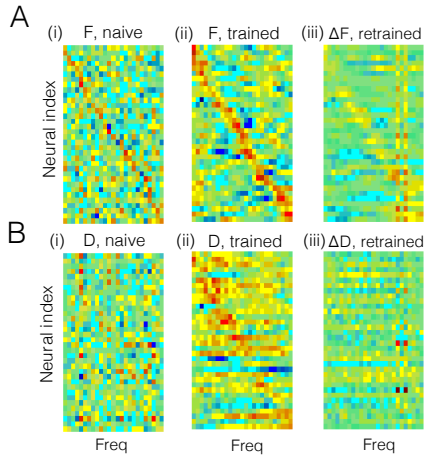


Figure 4.6: **Feedforward and recurrent connection structure before and after** (a1) Feedforward weights of neurons before learning, sorted according to maximal frequency. These weights correspond to the spectral receptive fields (SRF) of the neurons, since they weight the different frequency bands. Bluish colors correspond to negative values, reddish colors to positive values. (a2) Same as (a1), after learning. The SRFs now have an excitatory subfield, and one or two inhibitory subfields, compatible with SRFs observed in primary auditory cortex [86]. (a3) Change in SRFs after re-training with a new stimulus (see fig. 4.5d). The SRFs change selectively (positively and negatively) at the position of the trained frequencies. The frequency-selective change in SRFs is in line with fast plastic changes of SRFs observed following behavioral training [149]. There is also a small decrease in gain at other frequencies, due to the competition with the new stimulus. (b) Same as in (a), but for the decoding weights. (b1) Decoding weights of neurons (sorted as in a1) before learning appear random. (b2) After learning, the decoding weights are more structured and broader than the SRFs in (a2), compatible to the decoding filter of speech measured in auditory cortex [86]. Same sorting of neurons as in (a2). (b3) After re-training to the new stimulus, a small number of decoding filter (neurons) "specialize" to the new stimulus, while the decoding weights of the others change only mildly. The network thereby minimizes its firing rate response to the new stimulus, while still providing an accurate representation of it.

fields (fig. 4.6a1-2). Decoding weights are wider and more complex, thus exploiting the high correlations between frequency channels (fig. 4.6b1-2). These model predictions are broadly compatible with observations in the mammalian auditory pathway, and notably the representation of speech signals in A1 [86].

As a drawback, the network has become specialized, and a new "non-speech" stimulus results in poor EI balance, high firing rates, and poor coding (fig. 4.5d,e). After experiencing the new sound several times, however, the network represents the "non-speech" sound as precisely and parsimoniously as the previously experienced speech sounds (fig. 4.5f). After retraining to the new stimulus, feed-forward weights are modified specifically at the frequencies of the new stimulus (fig. 4.6a3). However, these changes are not massive. In particular, only a handful of neurons (two in this example) have become truly specialized to the new stimulus, as reflected by their decoding weights (fig. 4.6b3).

4.2.6 Robustness of Learning against perturbations

A crucial final question is whether these learning rules continue to work under more realistic conditions, such as noise in various components of the circuit, delays in the synaptic transmission, or constraints on the ability of arbitrary neurons to form synaptic connections in the first place. To answer these questions, we first note that the learning rules work independent of the initial state of the network. As long as the initial network dynamics are sufficiently stable, the learning rules converge globally (see Supplementary Text S9 and S10 for a proof). We furthermore note that the networks perform better and become more robust as the number of neurons increases (fig. 4.7a, see also [12]).

We first studied how the learning rules perform when not all neurons can form (potential) synaptic connections. As shown in fig. 4.7b, eliminat-

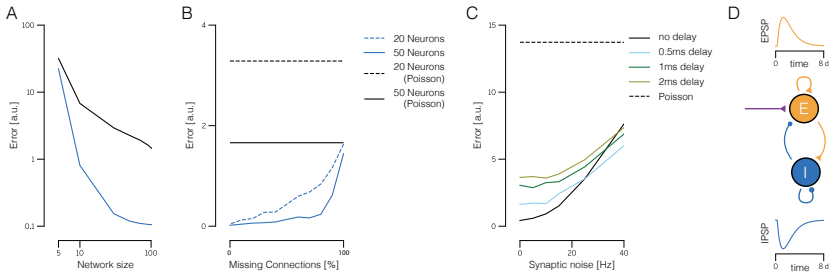


Figure 4.7: **Robustness of the learning rules to missing connections, noise, and synaptic delays. Simulations are based on EI networks receiving two-dimensional, random input signals. Network size is given as number of inhibitory neurons. The pool of excitatory neurons is twice as large.** (a) Performance (mean-square error between input signal and signal estimate) of the learnt network as a function of (inhibitory) network size. Trained network (blue) and equivalent Poisson rate network (black), given by neurons whose firing follows Poisson processes with identical average rates. (b) Performance of the learnt network as a function of connection sparsity. Here, we randomly deleted some percentage of the connections in the network, and then trained the remaining connections with the same learning rule as before. We adjusted the variance of the input signals to achieve the same mean firing rate in each neuron ($r = 5$ Hz in excitatory, $r = 10$ Hz in inhibitory neurons). Black lines denote the performance of an equivalent (and unconnected) population of Poisson-spiking neurons. (c) Network performance as a function of synaptic noise and synaptic delay. Here, we injected random white-noise currents into each neuron. The size of the noise was defined as the standard deviation of the injected currents, divided by the time constant and firing threshold. Roughly, this measure corresponds to the firing rate cause by the synaptic noise alone, in the absence of connections or input signals. As in B, the input variance was scaled to get the same mean firing rate in each neuron ($r = 5$ Hz in excitatory, $r = 10$ Hz in inhibitory neurons). Different colors show curves for different synaptic delays (see panel d). (d) Temporal profile of EPSCs and IPSCs (injected currents each time a spike is received) in the delayed networks, plotted as a function of the synaptic delay d. We rescaled the time axis to get the different delays used in panel c.

ing potential synapses only affects the performance of the learnt network when drastic limits are imposed (less than 20% of connections available for a network with $N = 50$ inhibitory neurons). Smaller networks are generally more sensitive (fig. 4.7b, dashed blue line), whereas larger networks are less sensitive.

To study the resistance of the learning rules against noise, we introduced random currents into the neurons, which can be viewed as a simulation of stochastic fluctuations in ion channels or background synaptic activity. For reasonable levels of noise, this modification had an essentially negligible effect on network performance. fig. 4.7c shows the error made by the networks after learning as a function of the strength of the introduced noise.

A final concern could be to what extent the learning rules rely on overtly simplistic synaptic dynamics—each spike causes a jump in the postsynaptic voltage followed by an exponential decay. To address this question, we also simulated the network assuming more realistic synaptic dynamics (fig. 4.7d). We measures the effective delay of transmission as the time-to-peak for a postsynaptic potential. Within the range of mono-synaptic transmission delays observed in cortical microcircuits, the networks still learns to encode their input signals efficiently, see fig. 4.7b. As transmission delays grow larger, a degradation in performance is incurred due to limited synchrony between similarly tuned neurons, which is unavoidable in the presence of delayed inhibition. Indeed, by keeping excitatory and inhibitory currents as balanced as possible, the network automatically finds an optimal regime of weak synchronization, removing the need for fine tuning of the network parameters. Such weak synchronization causes weak oscillations in the network activity whose time scales may be related to gamma rhythms [30].

Thus, we found that under a wide range of perturbations, the network learnt to achieve a performance near the discretization limit, outperform-

ing conventional spiking networks or population coding models based on Poisson spike trains. This robustness is inherited from the generality of the relationship between EI balance and the error-correcting coding strategy in the network [12, 19].

4.2.7 Manipulating plasticity

One of the key consequences of our derivations is that feedforward and recurrent plasticity serve different goals. Whereas recurrent plasticity works to balance the network, keeping all voltages (and thereby the respective coding errors) in check, feedforward plasticity works to unbalance each neuron, driving up excitation as much as possible. Since the recurrent plasticity rules are faster, they win this competition, and the network remains in a balanced state.

These considerations lead to some fundamental, yet experimentally testable predictions that are illustrated in fig. 4.8. In this simulated experiment, a number of neurons with similar tuning curves to angular stimuli (such as oriented gratings) are suddenly killed (fig. 4.8a, dashed arrows). In principle, this should severely impair the representation of stimuli in this direction. However, three mechanisms are recruited to compensate for the degradation of the representation. In a first step, the EI imbalance introduced by the lesion is immediately corrected by the network. This occurs instantaneously, before any plasticity mechanisms can be involved. As a result, the tuning curves of some neurons shift, widen, and increase in amplitude in an effort to cover the "hole" made in the representation (Figure 8B). This compensation is a result of instantaneous des-inhibition in the lesioned network, not plasticity [12]. While this re-balancing limits coding errors, it still leads to an inefficient representation due to the large firing rates required from compensating neurons. In a second step, the recurrent learning rules kick in, and the network adapts its recurrent connections so that each neuron is again balanced on a spike-by-spike time

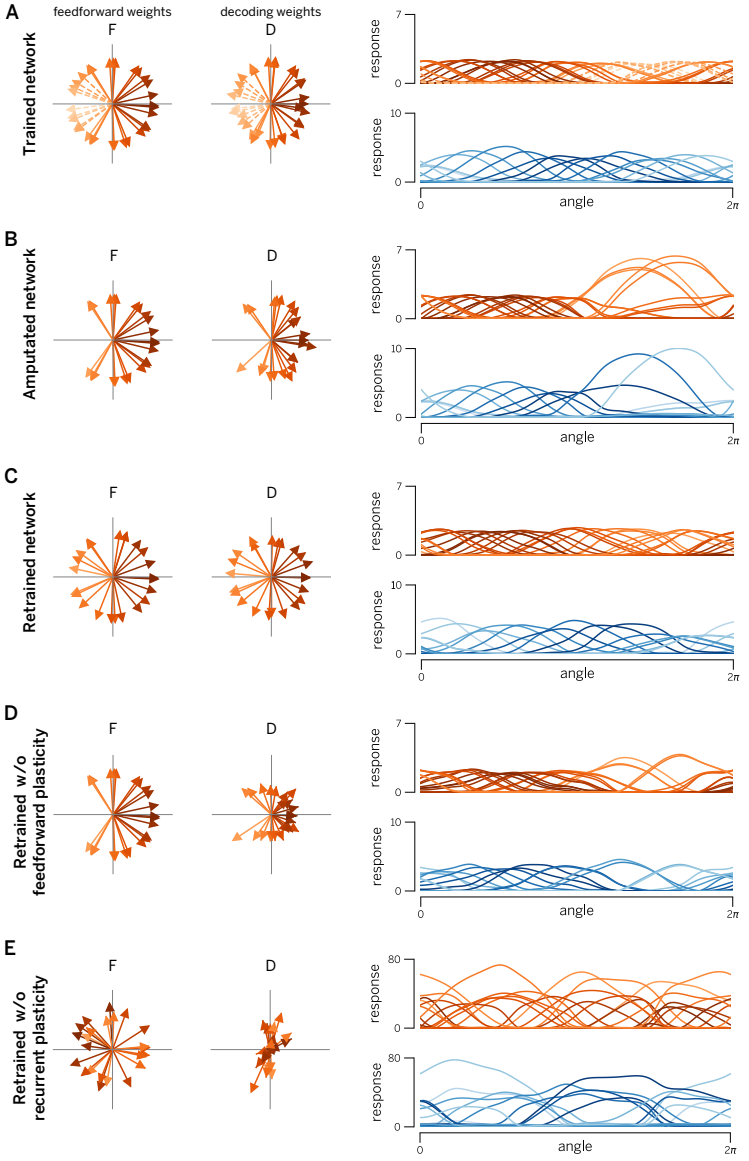


Figure 4.8: (legend on next page)

Figure 4.8: **Manipulating recurrent and feedforward plasticity.** (a) EI network with 80 excitatory neurons and 40 inhibitory neurons, trained with two uncorrelated time-varying inputs. Left panels: learnt feedforward weights of excitatory population. Central panels: Optimal decoding weights of excitatory population. Right panels: Tuning curves of excitatory neurons (red) and inhibitory neurons (blue). Neurons encoding/decodings weights and tuning curves are shaded according to their preferred direction (direction of their decoding weight vector) in 2D input space. The color code is maintained in all subsequent figures (even if their preferred direction changed after lesion and/or retraining). (b) Same network after deletion of leftward-coding excitatory neurons (see dashed lines in panel a). Note that no new training of the weights has yet taken place. Changes in tuning curves and decoding weights are due to internal network dynamics. We observe a large increase in firing rates and a widening and shifting of tuning curves towards the lesion, a signature that the network can still encode leftward moving stimuli, but does it in an inefficient way. (c) Retrained network. The lesioned network in (b) was subjected to 1000s re-training of the connection. Consequently, the "hole" induced by the lesion has been filled by the new feedforward/decoding weights, all tuning curves once again covering the input space uniformly. (d) Network with retrained recurrent connections (feedforward weights are the same as in panel b). Even without feedforward plasticity, the lesioned network is able to recover its efficiency to a large extent. (e) Network with retrained feedforward weights only (recurrent connections are the same as in panel b). While feedforward weights once again cover the input space, absence of recurrent plasticity results in a massive increase in firing rates and a concomitant decrease in coding precision (not shown). Consequently, training only feedforward weights after a lesion actually worsens the representation.

scale. In a third step, the network also re-learns the feedforward weights through the slower feedforward learning rules. As a consequence, the final, adapted network again covers the input space uniformly, just with fewer neurons (fig. 4.8c).

Importantly, and as already shown in fig. 4.2, plasticity of the recurrent EI loop (including E to I connections) is more important for this process than plasticity on the feedforward weights. This observation leads to the following prediction: even in the absence of feedforward plasticity, the network recovers most of its efficiency (fig. 4.8d). While the tuning curves never achieve the perfect re-arrangement of the network with intact plasticity (fig. 4.8c), the responses of overactive cells are suppressed and shifted further towards the impaired direction. In contrast, if we were to block recurrent plasticity (fig. 4.8e), the network would become unbalanced and thereby inefficient due to the remaining action of the feedforward weights. While selectively blocking plasticity mechanisms at different synapses may seem a bit outlandish at first, the modern molecular biology toolbox does put it within reach [89]. By combining such techniques with focal lesions and awake recordings (e.g. calcium imaging) in local neural populations, these predictions are now within the range of the testable.

A second important prediction arises from the differential time course of feedforward and recurrent plasticity. Since recurrent plasticity should be much faster than feedforward plasticity, we predict that a partial recovery of the network efficiency will occur relatively fast (in minutes to hours of exposure to the stimuli with orientations matching the knocked-out cells). This will be performed mainly through a re-equalization of the population responses, but without major changes in the preferred tuning of the cells (compare fig. 4.8b and fig. 4.8d). It will eventually be followed by a slower recovery of the tuning curve shapes and uniform density (but at a much slower time scale, e.g. over days or weeks of exposure).

4.3 Discussion

In summary, we have shown how populations of excitatory and inhibitory neurons can learn to efficiently represent a set of signals spike by spike. We have measured efficiency with an objective function that combines the mean-square reconstruction error with various cost terms. While mathematically simpler than mutual-information-based approaches, our objective function includes both principal and overcomplete independent component analysis as special cases [76, 61]. This type of unsupervised learning has previously been studied extensively in rate networks [94, 78, 14, 5, 79, 139, 98, 99, 63]. Implementations that seek to mimic biology by assuming spiking neurons, recurrent network architectures, and local learning rules, have always faced difficulties, and have therefore been largely limited to heuristic or approximate approaches, [110, 34, 151, 68, 27]. Using a rigorous, spike-based, and top-down approach, we have here derived biologically plausible learning rules that are guaranteed to converge to a specific connectivity and achieve a maximally efficient spike code. Importantly, single spikes are not to be considered as random samples from a rate, but are rather an integral part of a metabolically efficient brain.

We have limited our study on learning here to the encoding of time-varying signals into spikes. Several questions seem natural at this point. First, beyond peripheral sensory systems, most neurons receive spikes as inputs, not analog signals, which seems to violate one of our core premises. Second, neural systems perform computations with the signals they receive, rather than encoding them into spikes, only to be read out again at a later stage, which may seem a rather pointless exercise. Third, our learning rules have been derived in an unsupervised scenario, and one may wonder whether the core ideas underlying these rules can be extended beyond that.

Concerning spiking inputs, we note that nothing prevents us from replacing the analog input signals with spike trains. While we have chosen to

explain these learning rules using analog inputs, our derivations were not dependent on this restriction. In fact, we have already used spike trains rather than analog input signals in the simulation of the EI-network in fig. 4.4—here the inhibitory neurons received spike trains from the excitatory neurons as 'feedforward' inputs, and we applied exactly the same feedforward learning rules as for the continuous-valued input signals (see also Supplementary Text S13).

Concerning computations, we note that the solution to the encoding problem provides a necessary starting point for introducing more complex computations. For example, we showed previously that a second set of slower connections can implement arbitrary linear dynamics in optimally designed networks [19]. Non-linear computations can be introduced as well, but require that these non-linearities are implemented in synapses or dendrites [132]. The separation between coding and computation in these approaches is very similar to the separation used in the neural engineering framework [41].

Concerning learning, we note that there has been quite a lot of progress in recent years in developing local learning rules in supervised scenarios, both in feedforward [52, 142, 107] and recurrent networks [50, 2]. We believe that the framework presented here provides crucial intuitions for supervised learning in spiking networks, since it shows how to represent global errors in local quantities such as voltages. In the future, these ideas may be combined with explicit singe-neuron models [135] to turn local learning rules into global functions [37, 2, 107].

Apart from the theoretical advances in studying learning in spiking networks, many of the critical features that are hallmarks of cortical dynamics follow naturally from our framework, even though they were not included in the original objective. We list four of the most important features. First, the predicted spike trains are highly irregular and variable, which has indeed been widely reported in cortical neurons [134, 147]. Im-

portantly, this variability is a signature of the network's coding efficiency, rather than detrimental [116] or purposeful noise [46, 25]. Second, despite this spike train variability, the membrane potentials of similarly tuned neurons are strongly correlated (due to shared inputs), as has indeed been found in various sensory areas [101, 150]. Third, local and recurrent inhibition in our network serves to balance the excitatory feedforward inputs on a very fast time scale. Such EI balance, in which inhibitory currents track excitatory currents on a millisecond time scale has been found in various systems and under various conditions [62, 148]. Fourth, we have derived learning rules whose polarity depends on the relative timing of pre-and postsynaptic spikes (see insets in fig. 4.4a). In fact, the respective sign switches simply reflect the immediate sign reversal of the coding error (and thus of the membrane potential) after each new spike. As a result, even though our proposed learning rules are not defined in terms of relative timing of pre- and postsynaptic spikes, most connections display some features of the classic STDP rules, e.g., LTP for pre-post pairing, and LTD for post-pre pairing [28, 45]. The only exception are E-E connections that exhibit "reverse STDP", i.e. potentiation for post-pre pairing (fig. 4.4a). Despite their simplicity, these rules are not only spike-time dependent but also weight and voltage-dependent, as observed experimentally [34].

Our framework thereby bridges from the essential biophysical quantities, such as the membrane voltages of the neurons, to the resulting population code, while providing crucial new insights on learning and coding in spiking neural networks.

4.4 Materials and methods

Detailed mathematical derivations of the learning rules are explained in the supplementary materials, available online at the following URL:

https://journals.plos.org/ploscompbiol/article/file?id=10.1371/journal.pcbi.1007692.s001&type=supplementary.
In addition, MATLAB code for the key simulations of the article is available at https://github.com/machenslab/spikes.

4.5 Author contributions

R.B. and S.D. derived and implemented biologically plausible learning rules, whereas W.B., P.V., and C.K.M. derived and implemented optimal learning rules. C.K.M. and S.D. drafted the main text, W.B. and P.V. drafted the Supplementary Materials. All authors reviewed and edited the paper.

Chapter 5

General discussion

This research aimed to tackle the computational problem of inferring relevant variables that are not directly observable. We proceeded with a multi-pronged approach.

5.1 Decision-making under uncertainty

The inference of latent variables is a pervasive computational problem in naturalistic animal behavior. Therefore, we reasoned, both simpler and more complex organisms must have developed mechanisms to tackle it.

By means of a carefully designed cross-species task, we tested, so to speak, "nature's solution" to this problem. We discovered that mice can learn the hidden structure of the environment, on reasonably fast timescales. As the task design allows us to quantitatively disambiguate optimal inference-based strategies from suboptimal value-based heuristics, we could monitor the "inference performance" of mice, and saw that it plateaued to an optimum after a few sessions. Moreover, mice were able to adjust their behavior to task parameters, whether explicit (travel cost) or implicit (probability values), consistently with the behavior of an optimal agent.

As an added advantage of our computationally-oriented approach, we were able to translate *verbatim* the rodent behavioral task into a video game for humans. There, the same normative predictions that held true for mice were again verified, with a notable difference. Whereas mice took several sessions to learn the structure of the task, humans could do so in only a few trials. This result, however, is potentially confounded by any preconceived notion that human subjects could have about the task, as well as by the instructions they received.

Having established that mice and humans perform similar computations in analogous tasks, we took advantage of modern genetic tools in rodents to determine the underlying mechanisms. We inactivated optogenetically two prefrontal cortical regions—Anterior Cingulate Cortex (ACC) and Orbito-Frontal Cortex (OFC)—in an interleaved fashion while mice were performing the task.

Using optogenetics, we showed that ACC and OFC contribute differentially to this computational process. In particular, whereas ACC inactivation corresponded to an increase in patience, but did not affect the inference process, OFC inactivation disrupted the optimal inference process and caused mice to revert to a simpler value-based strategy.

5.2 Intelligible artificial intelligence

We addressed the latent variable decoding problem from a normative perspective, using contemporary tools from machine learning. There, we aimed for generality. Even though specific tasks used in neuroscience labs often admit explicit optimal solutions, real-world problems often do not: as the problem complexity increases, optimal solutions become intractable. The core of the computation consists in finding an expressive, differentiable parameterization of the solution space. An "optimal" solution (technically,

a local minimum of the loss function) can then be found with standard gradient-based optimization algorithms.

In the *parametric machine* framework, we were able to show that such efficient parameterizations can be characterized in terms of the existence and uniqueness of the solution of a simple equation. We explored three different families of parametric machines, based on hypergraph, integral equations, and kernels.

We formalized the notion of feedforward neural network in terms of hypergraph representations and showed that the global transformation (i.e., the sum of all layers) of a feedforward neural network respects the machine property. This represented our prototypical *discrete architecture*. Dissatisfied with the manual choice of a hypergraph to encode the architecture, we borrowed ideas from kernel methods and showed that, in the case of small datasets, there is no need to select an architecture manually. Instead, all connections can be allowed, without incurring an uncontrolled increase in the number of parameters. This is a classical trick in the framework or kernel Ridge regression, but to our knowledge, it had not yet been used to define multilayer architectures with shortcut connections.

We implemented hypergraph-based and kernel machines in PyTorch [96]. Using automatic differentiation software, we computed their derivatives with respect to a differentiable loss function. We trained kernel machines alongside deep neural networks with a comparable number of parameters and showed that the kernel machine reaches higher accuracies and interpolates more smoothly on small datasets.

A new line of research in machine learning [32] argues that some neural networks can be reframed in terms of ordinary differential equations (ODEs). Indeed, computing the forward pass of a *residual network* [55] is equivalent to solving a discretized ODE using the Euler method. It could, therefore, be more efficient to drop the discretization and adopt adaptive algorithms. This novel approach goes under the name

of *Neural ODE*: it only requires learning a single function $\phi(u, t)$ that encodes a derivative, and decouples the definition of the problem (the function ϕ) from its computation (the choice of an ODE solver). Intrigued by the conceptual simplicity of Neural ODEs, we investigated whether such continuous architectures were also a particular case of the machine framework. Again, we found a positive answer: an ODE can be framed as a nonlinear Volterra integral equation of the second kind, which satisfies the machine characterization.

Combining kernel machines and known results on efficient methods to solve Volterra equations of a specific type, we were able to define *infinite-depth kernel machines*. We implemented infinite-depth kernel machines in PyTorch, and tested their performance on an image classification task with a tiny training dataset (one example per class).

5.3 Biologically plausible learning

Massively distributed computing systems, consisting of many interconnected simple units, were put forward as a model of cognition several decades ago [121]. Arguably, modern neural networks, and more generally parametric machines, respect the fundamental hypotheses of the connectionist *credo*. In recent years, this is in no small part due to technical rather than conceptual reasons: connectionist architectures, due to their parallel nature, are particularly well-suited for Graphics Processing Units [59].

Nonetheless, from a neuroscientific perspective, it remains interesting to verify whether neural tissue is a suitable computing device for connectionist architectures. While the elementary computations of artificial neurons—linear combinations and pointwise nonlinearities—are certainly within the capabilities of biological neurons and synapses, the situation is more delicate in the case of learning. Even if the network is *sparsely connected*, the optimal learning rate, derived mathematically from an ob-

jective function, will depend on *global* quantities that are not available *locally* to each synapse.

Here, we focused on a specific computation—encoding and decoding efficiently and robustly a low-dimensional analog signal. We devised simple plasticity rules for feedforward and recurrent synapses, based on postsynaptic voltages and currents at the time of a presynaptic spike. The recurrent rules aimed to tighten the excitation/inhibition balance of the network on a very fast timescale, on the order of a single population interspike interval.

In general, the projections of the input signal onto the cells of an encoding population have no reason to be equal to the decoding weights, yet we showed that this is optimal in the case of whitened input, and devised feedforward rules to achieve this. Furthermore, we discussed how to modify the learning rules, preserving the locality constraints, in the case of non-white input.

Even though the computation we studied—encoding and decoding a signal—may seem trivial, we find that it can be beneficial for several reasons. On the one hand, in the presence of regularization costs, our network can compute *sparse, overcomplete bases* [76]. We showed that the network can learn correlation in the structure of speech signals, in order to represent them with high precision. On the other hand, by encoding a low-dimensional signal in a large network, it is possible to gain robustness. In particular, it has been shown that our network architecture is resilient to neural death: as soon as a cell fails, remaining neurons with overlapping receptive fields compensate for it instantaneously and optimally. Indeed, we tested robustness to a variety of issues—missing connections, noise, and synaptic delays—and showed that the network's performance degraded gracefully.

Finally, to bring our theoretical network closer to experimental data, we modified it to include two separate populations of excitatory and in-

hibitory cells. In this scenario, we tested how our plasticity rules would respond to experimental protocols designed to probe synaptic potentiation and depotentiation mechanisms.

5.4 Future directions

This thesis discussed the general problem of decoding relevant latent variables from several perspectives: behavioral neuroscience, artificial intelligence, and theoretical neuroscience. While these three approaches were so far pursued separately, it is interesting to speculate on how they could be combined.

Artificial versus biological agents in video games. In chapter 2, we prioritized simplicity in designing our task, both in its rodent and human versions. Yet, especially in the case of the human video game, it is natural to imagine a more complex, engaging version. For instance, in classical Atari games, artificial agents were only recently able to achieve performance comparable with humans [88]. The key to the excellent performance of human experts may not be dexterity, but rather the ability to infer and predict future events. Experienced players could for example learn the statistical structure of the enemies' attacks, and make use of this knowledge in their strategy. In the future, I plan to design a simple, but engaging, *bullet hell* video game (see fig. 5.1 for a preview), and make it available online to collect a large sample of playing data. Hopefully, such a dataset will make it possible to uncover the inference processes underlying the players' decisions. Qualitative and quantitative comparisons with such a dataset could be a useful benchmark for artificial agents (e.g., parametric machines) in Reinforcement Learning. In particular, it will be interesting to correlate the dimensionality of the *hidden space* of the video game—

how many hidden variables are sufficient statistics of the game—with the performance of both human and artificial agents.

Figure 5.1: A prototype of an engaging inference-based video game.

Efficient relevant coding. Both the parametric machines and the spiking neural network models of efficient coding, developed in chapters 3 and 4 respectively, are unfortunately not optimized for Reinforcement Learning. This is not an intrinsic limitation of the models, but rather a consequence of the choice of the loss function. The artificial agents aim to encode *all* information they receive, whether it is relevant to solve a given task or not. This is particularly problematic, as recurrent neural networks tend to have limited memory, especially in the case of high-dimensional inputs. In the future, I plan to investigate how classical loss functions can be modified to take into account the relevance of information in the context of a task.

Biologically plausible machines. The biologically plausible learning rules discussed in chapter 4 are limited to specific computations—encoding

and decoding. The network structure can be extended with slow recurrent connections to track arbitrary dynamical systems, as shown in [19]. This more complex architecture can be learned with local learning rules [2]. However, it will be interesting to explore whether this approach generalizes to deeper architectures. In particular, I speculate that a biological implementation of finite- and infinite-depth parametric machines may require in general feedback connections, to mimic the backward pass of reverse-mode differentiation [125]. The error signal, received in the final stages of the computation, needs to be backpropagated to the early processing steps, in order to optimize those. It will be interesting to investigate to what extent the microcircuitry of feedback connections in the neocortex can be understood under the light of reverse-mode differentiation.

Bibliography

[1] M. Akrout, C. Wilson, P. Humphreys, T. Lillicrap, and D. B. Tweed. Deep learning without weight transport. In *Advances in Neural Information Processing Systems*, pages 974–982, 2019.

[2] A. Alemi, C. K. Machens, S. Deneve, and J.-J. Slotine. Learning nonlinear dynamics in efficient, balanced spiking networks using local plasticity rules. In *Thirty-Second AAAI Conference on Artificial Intelligence*, pages 588–595, 2018.

[3] M. A. Álvarez, L. Rosasco, and N. D. Lawrence. Kernels for Vector-Valued Functions: A Review. *Foundations and Trends® in Machine Learning*, 4(3):195–266, June 2012.

[4] S.-i. Amari. Dynamics of pattern formation in lateral-inhibition type neural fields. *Biological cybernetics*, 27(2):77–87, 1977.

[5] S.-i. Amari, A. Cichocki, and H. H. Yang. A new learning algorithm for blind signal separation. In *Advances in neural information processing systems*, pages 757–763, 1996.

[6] D. J. Amit and N. Brunel. Model of global spontaneous activity and local structured activity during delay periods in the cerebral cortex. *Cerebral cortex*, 7(3):237–252, 1997.

[7] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3):337–404, 1950.

[8] K. J. Astrom. Optimal control of markov decision processes with incomplete state estimation. *J. Math. Anal. Applic.*, 10:174–205, 1965.

[9] H. Attias. Planning by probabilistic inference. In *AISTATS*. Citeseer, 2003.

[10] B. B. Averbeck, P. E. Latham, and A. Pouget. Neural correlations, population coding and computation. *Nature Reviews Neuroscience*, 7(5):358–366, 2006.

[11] C. T. H. Baker. A perspective on the numerical treatment of Volterra equations. *Journal of Computational and Applied Mathematics*, 125(1):217–249, Dec. 2000.

[12] D. T. Barrett, S. Denève, and C. K. Machens. Optimal compensation for neuron loss. *eLife*, 5:e12454, 2016.

[13] D. Bates, José Bayoán Santiago Calderón, D. Kleinschmidt, T. Kelman, S. Babayan, P. K. Mogensen, M. Piibeleht, M. Bouchet-Valat, M. Hatherly, E. Saba, A. Baldassari, and A. Noack. Dmbates/MixedModels.jl: Avoid fallback to generic_matmul, Mar. 2019.

[14] A. J. Bell and T. J. Sejnowski. An information-maximization approach to blind separation and blind deconvolution. *Neural computation*, 7(6):1129–1159, 1995.

[15] R. Ben-Yishai, R. L. Bar-Or, and H. Sompolinsky. Theory of orientation tuning in visual cortex. *Proceedings of the National Academy of Sciences*, 92(9):3844–3848, 1995.

[16] M. G. Bergomi, P. Frosini, D. Giorgi, and N. Quercioli. Towards a topological–geometrical theory of group equivariant non-expansive operators for data analysis and machine learning. *Nature Machine Intelligence*, pages 1–11, Sept. 2019.

[17] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah. Julia: A Fresh Approach to Numerical Computing. *SIAM Review*, 59(1):65–98, Jan. 2017.

[18] C. M. Bishop and P. o. N. C. C. M. Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Nov. 1995.

[19] M. Boerlin, C. K. Machens, and S. Denève. Predictive coding of dynamical variables in balanced spiking networks. *Plos Computiational Biology*, 9(11):e1003258, 2013.

[20] R. Bourdoukan, D. Barrett, C. K. Machens, and S. Deneve. Learning optimal spike-based representations. In *Advances in neural information processing systems*, pages 2285–2293, 2012.

[21] J. M. Bownds. Theory and performance of a subroutine for solving Volterra Integral Equations. *Computing*, 28(4):317–332, Dec. 1982.

[22] X. Boyen, N. Friedman, and D. Koller. Discovering the Hidden Structure of Complex Dynamic Systems. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence*, UAI'99, pages 91–100, Stockholm, Sweden, 1999. Morgan Kaufmann Publishers Inc.

[23] D. A. Braun, C. Mehring, and D. M. Wolpert. Structure learning in action. *Behavioural Brain Research*, 206(2):157–165, Jan. 2010.

[24] B. W. Brunton, M. M. Botvinick, and C. D. Brody. Rats and Humans Can Optimally Accumulate Evidence for Decision-Making. *Science*, 340(6128):95–98, Apr. 2013.

[25] L. Buesing, J. Bill, B. Nessler, and W. Maass. Neural dynamics as sampling: a model for stochastic computation in recurrent networks of spiking neurons. *PLoS Comput Biol*, 7(11):e1002211, 2011.

[26] Y. Burak and I. R. Fiete. Fundamental limits on persistent activity in networks of noisy neurons. *Proceedings of the National Academy of Sciences*, 109(43):17645–17650, 2012.

[27] K. S. Burbank. Mirrored stdp implements autoencoder learning in a network of spiking neurons. *PLoS computational biology*, 11(12):e1004566, 2015.

[28] N. Caporale and Y. Dan. Spike timing-dependent plasticity: a hebbian learning rule. *Annu. Rev. Neurosci.*, 31:25–46, 2008.

[29] A. R. Cassandra, L. P. Kaelbling, and M. L. Littman. Acting optimally in partially observable stochastic domains. In *AAAI*, volume 94, pages 1023–1028, 1994.

[30] M. Chalk, B. Gutkin, and S. Deneve. Neural oscillations as a signature of efficient coding in the presence of synaptic delays. *Elife*, 5, 2016.

[31] E. L. Charnov. Optimal foraging, the marginal value theorem. *Theoretical Population Biology*, 9(2):129–136, Apr. 1976.

[32] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. K. Duvenaud. Neural Ordinary Differential Equations. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems 31*, pages 6571–6583. Curran Associates, Inc., 2018.

[33] Y. Cho and L. K. Saul. Kernel Methods for Deep Learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 342–350. Curran Associates, Inc., 2009.

[34] C. Clopath, L. Büsing, E. Vasilaki, and W. Gerstner. Connectivity reflects coding: a model of voltage-based stdp with homeostasis. *Nature neuroscience*, 13(3):344–352, 2010.

[35] N. D. Daw, S. J. Gershman, B. Seymour, P. Dayan, and R. J. Dolan. Model-based influences on humans' choices and striatal prediction errors. *Neuron*, 69(6):1204–1215, Mar. 2011.

[36] N. D. Daw, Y. Niv, and P. Dayan. Uncertainty-based competition between prefrontal and dorsolateral striatal systems for behavioral control. *Nature Neuroscience*, 8(12):1704–1711, Dec. 2005.

[37] S. Denève, A. Alemi, and R. Bourdoukan. The brain as an efficient and robust adaptive learner. *Neuron*, 94(5):969–977, 2017.

[38] S. Denève and C. K. Machens. Efficient codes and balanced networks. *Nature neuroscience*, 19(3):375–382, 2016.

[39] L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

[40] P. Drineas and M. W. Mahoney. On the Nyström Method for Approximating a Gram Matrix for Improved Kernel-Based Learning. *Journal of Machine Learning Research*, 6(Dec):2153–2175, 2005.

[41] C. Eliasmith. A unified approach to building and controlling spiking attractor networks. *Neural computation*, 17(6):1276–1314, 2005.

[42] C. Eliasmith, T. C. Stewart, X. Choo, T. Bekolay, T. DeWolf, Y. Tang, and D. Rasmussen. A large-scale model of the functioning brain. *science*, 338(6111):1202–1205, 2012.

[43] T. Elsken, J. H. Metzen, and F. Hutter. Neural architecture search: A survey. *arXiv preprint arXiv:1808.05377*, 2018.

[44] N. Eshel, J. Tian, M. Bukwich, and N. Uchida. Dopamine neurons share common response function for reward prediction error. *Nature Neuroscience*, 19(3):479–486, Mar. 2016.

[45] D. E. Feldman. The spike-timing dependence of plasticity. *Neuron*, 75(4):556–571, 2012.

[46] J. Fiser, P. Berkes, G. Orbán, and M. Lengyel. Statistically optimal perception and learning: from behavior to neural representations. *Trends in cognitive sciences*, 14(3):119–130, 2010.

[47] G. Gallo, G. Longo, S. Pallottino, and S. Nguyen. Directed hypergraphs and applications. *Discrete Applied Mathematics*, 42(2):177–201, Apr. 1993.

[48] D. Ganguli and E. Simoncelli. Efficient sensory encoding and bayesian inference with heterogeneous neural populations. *Neural Computation*, 26(10):2103–34, 2014.

[49] A. P. Georgopoulos, A. B. Schwartz, and R. E. Kettner. Neuronal population coding of movement direction. *Science*, 233(4771):1416–1419, 1986.

[50] A. Gilra and W. Gerstner. Predicting non-linear dynamics by stable local learning in a recurrent spiking neural network. *Elife*, 6:e28295, 2017.

[51] J. I. Gold and M. N. Shadlen. The Neural Basis of Decision Making. *Annual Review of Neuroscience*, 30(1):535–574, 2007.

[52] J. Guerguiev, T. P. Lillicrap, and B. A. Richards. Towards deep learning with segregated dendrites. *ELife*, 6:e22901, 2017.

[53] M. Hausknecht and P. Stone. Deep recurrent q-learning for partially observable mdps. In *2015 AAAI Fall Symposium Series*, 2015.

[54] B. Y. Hayden, J. M. Pearson, and M. L. Platt. Neuronal basis of sequential foraging decisions in a patchy environment. *Nature Neuroscience*, 14(7):933–939, July 2011.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[56] G. Hennequin, E. J. Agnes, and T. P. Vogels. Inhibitory plasticity: balance, control, and codependence. *Annual Review of Neuroscience*, 40:557–579, 2017.

[57] R. J. Herrnstein. Relative and Absolute Strength of Response as a Function of Frequency of Reinforcement1,2. *Journal of the Experimental Analysis of Behavior*, 4(3):267–272, 1961.

[58] T. Hofmann, B. Schölkopf, and A. J. Smola. Kernel Methods in Machine Learning. *The Annals of Statistics*, 36(3):1171–1220, 2008.

[59] S. Hong and H. Kim. An analytical model for a gpu architecture with memory-level and thread-level parallelism awareness. In *Proceedings of the 36th annual international symposium on Computer architecture*, pages 152–163, 2009.

[60] D. H. Hubel and T. N. Wiesel. Receptive fields, binocular interaction and functional architecture in the cat's visual cortex. *The Journal of physiology*, 160(1):106–154, 1962.

[61] A. Hyvärinen, J. Karhunen, and E. Oja. *Independent component analysis*, volume 46. John Wiley & Sons, 2004.

[62] J. S. Isaacson and M. Scanziani. How inhibition shapes cortical activity. *Neuron*, 72(2):231–243, 2011.

[63] T. Isomura and T. Toyoizumi. A local learning rule for independent component analysis. *Scientific reports*, 6:28073, 2016.

[64] H. Kadri, E. Duflos, P. Preux, S. Canu, A. Rakotomamonjy, and J. Audiffren. Operator-valued kernels for learning from functional response data. *The Journal of Machine Learning Research*, 17(1):613–666, 2016.

[65] L. V. Kantorovich and G. P. Akilov. *Functional Analysis*. Pergamon Press, Oxford ; New York, 2d ed edition, 1982.

[66] T. Kawai, H. Yamada, N. Sato, M. Takada, and M. Matsumoto. Roles of the Lateral Habenula and Anterior Cingulate Cortex in Negative Outcome Monitoring and Behavioral Adjustment in Nonhuman Primates. *Neuron*, 88(4):792–804, Nov. 2015.

[67] G. Kimeldorf and G. Wahba. Some results on tchebycheffian spline functions. *Journal of mathematical analysis and applications*, 33(1):82–95, 1971.

[68] P. D. King, J. Zylberberg, and M. R. DeWeese. Inhibitory interneurons decorrelate excitatory cells to drive sparse code formation in a spiking model of v1. *The Journal of Neuroscience*, 33(13):5475–5485, 2013.

[69] N. Kolling, T. E. J. Behrens, R. B. Mars, and M. F. S. Rushworth. Neural Mechanisms of Foraging. *Science*, 336(6077):95–98, Apr. 2012.

[70] N. Kolling, M. Wittmann, and M. F. S. Rushworth. Multiple Neural Mechanisms of Decision Making and Their Competition under Changing Risk Pressure. *Neuron*, 81(5):1190–1202, Mar. 2014.

[71] N. Kolling, M. K. Wittmann, T. E. J. Behrens, E. D. Boorman, R. B. Mars, and M. F. S. Rushworth. Value, search, persistence and model updating in anterior cingulate cortex. *Nature Neuroscience*, 19(10):1280–1285, Oct. 2016.

[72] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, May 2017.

[73] B. J. Lansdell, P. Prakash, and K. P. Kording. Learning to solve the credit assignment problem. *arXiv preprint arXiv:1906.00889*, 2019.

[74] J. Lee, Y. Bahri, R. Novak, S. S. Schoenholz, J. Pennington, and J. Sohl-Dickstein. Deep Neural Networks as Gaussian Processes. *arXiv:1711.00165 [cs, stat]*, Oct. 2017.

[75] E. S. Lein, M. J. Hawrylycz, N. Ao, M. Ayres, A. Bensinger, A. Bernard, A. F. Boe, M. S. Boguski, K. S. Brockway, E. J. Byrnes, L. Chen, L. Chen, T.-M. Chen, M. Chi Chin, J. Chong, B. E. Crook, A. Czaplinska, C. N. Dang, S. Datta, N. R. Dee, A. L. Desaki, T. Desta, E. Diep, T. A. Dolbeare, M. J. Donelan, H.-W. Dong, J. G. Dougherty, B. J. Duncan, A. J. Ebbert, G. Eichele, L. K. Estin, C. Faber, B. A. Facer, R. Fields, S. R. Fischer, T. P. Fliss, C. Frensley, S. N. Gates, K. J. Glattfelder, K. R. Halverson, M. R. Hart, J. G. Hohmann, M. P. Howell, D. P. Jeung, R. A. Johnson, P. T. Karr, R. Kawal, J. M. Kidney, R. H. Knapik, C. L. Kuan, J. H. Lake, A. R. Laramee, K. D. Larsen, C. Lau, T. A. Lemon, A. J. Liang, Y. Liu, L. T. Luong, J. Michaels, J. J. Morgan, R. J. Morgan, M. T. Mortrud, N. F. Mosqueda, L. L. Ng, R. Ng, G. J. Orta, C. C. Overly, T. H. Pak, S. E. Parry, S. D. Pathak, O. C. Pearson, R. B. Puchalski, Z. L. Riley, H. R. Rockett, S. A. Rowland, J. J. Royall, M. J. Ruiz, N. R. Sarno, K. Schaffnit, N. V. Shapovalova, T. Sivisay, C. R. Slaughterbeck, S. C. Smith, K. A. Smith, B. I. Smith, A. J. Sodt, N. N. Stewart, K.-R. Stumpf, S. M. Sunkin, M. Sutram, A. Tam, C. D. Teemer, C. Thaller, C. L. Thompson, L. R. Varnam, A. Visel, R. M. Whitlock, P. E. Wohnoutka, C. K. Wolkey, V. Y. Wong, M. Wood, M. B. Yaylaoglu, R. C. Young, B. L. Youngstrom, X. Feng Yuan, B. Zhang, T. A. Zwingman, and A. R.

Jones. Genome-wide atlas of gene expression in the adult mouse brain. *Nature*, 445(7124):168–176, Jan. 2007.

[76] M. S. Lewicki and T. J. Sejnowski. Learning overcomplete representations. *Neural computation*, 12(2):337–365, 2000.

[77] X. Li, H. Chen, X. Qi, Q. Dou, C.-W. Fu, and P.-A. Heng. H-DenseUNet: Hybrid Densely Connected UNet for Liver and Tumor Segmentation From CT Volumes. *IEEE Transactions on Medical Imaging*, 37(12):2663–2674, Dec. 2018.

[78] R. Linsker. Self-organization in a perceptual network. *Computer*, 21(3):105–117, 1988.

[79] R. Linsker. A local learning rule that enables information maximization for arbitrary input distributions. *Neural Computation*, 9(8):1661–1665, 1997.

[80] M. L. Littman. The witness algorithm: Solving partially observable markov decision processes. *Brown University, Providence, RI*, 1994.

[81] E. Lottem, D. Banerjee, P. Vertechi, D. Sarra, M. oude Lohuis, and Z. F. Mainen. Activation of serotonin neurons promotes active persistence in a probabilistic foraging task. *Nature Communications*, 9(1), Dec. 2018.

[82] S. MacLane. *Categories for the Working Mathematician*. Springer Science & Business Media, Apr. 2013.

[83] J. Mairal. End-to-End Kernel Learning with Supervised Convolutional Kernel Networks. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1399–1407. Curran Associates, Inc., 2016.

[84] J. Mairal, P. Koniusz, Z. Harchaoui, and C. Schmid. Convolutional Kernel Networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2627–2635. Curran Associates, Inc., 2014.

[85] R. A. McLean, W. L. Sanders, and W. W. Stroup. A Unified Approach to Mixed Linear Models. *The American Statistician*, 45(1):54–64, 1991.

[86] N. Mesgarani, S. David, J. Fritz, and S. Shamma. Mechanisms of noise robust representation of speech in primary auditory cortex. *Proc Natl Acad Sci*, 111(18):6792–7, 2014.

[87] C. A. Micchelli and M. Pontil. On learning vector-valued functions. *Neural computation*, 17(1):177–204, 2005.

[88] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[89] H. Murakoshi, M. E. Shin, P. Parra-Bueno, E. M. Szatmari, A. C. Shibata, and R. Yasuda. Kinetics of endogenous camkii required for synaptic plasticity revealed by optogenetic kinase inhibitor. *Neuron*, 94(1):37–47, 2017.

[90] N. S. Narayanan, J. F. Cavanagh, M. J. Frank, and M. Laubach. Common medial frontal mechanisms of adaptive control in humans and rodents. *Nature Neuroscience*, 16(12):1888–1895, Dec. 2013.

[91] R. M. Neal. Priors for Infinite Networks. In R. M. Neal, editor, *Bayesian Learning for Neural Networks*, Lecture Notes in Statistics, pages 29–53. Springer New York, New York, NY, 1996.

[92] Y. Niv. Learning task-state representations. *Nature Neuroscience*, 22(10):1544–1553, Oct. 2019.

[93] Y. Niv, R. Daniel, A. Geana, S. J. Gershman, Y. C. Leong, A. Radulescu, and R. C. Wilson. Reinforcement Learning in Multidimensional Environments Relies on Attention Mechanisms. *Journal of Neuroscience*, 35(21):8145–8157, May 2015.

[94] E. Oja. Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273, 1982.

[95] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381(6583):607, 1996.

[96] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer. Automatic differentiation in PyTorch. Oct. 2017.

[97] J. Pearl. Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. *Journal of Philosophy*, 88(8):434–437, 1991.

[98] C. Pehlevan and D. Chklovskii. A normative theory of adaptive dimensionality reduction in neural networks. In *Advances in Neural Information Processing Systems*, pages 2269–2277, 2015.

[99] C. Pehlevan, T. Hu, and D. B. Chklovskii. A hebbian/anti-hebbian neural network for linear subspace learning: A derivation from multidimensional scaling of streaming data. *Neural computation*, 27(7):1461–1495, 2015.

[100] C. Pehlevan, A. M. Sengupta, and D. B. Chklovskii. Why do similarity matching objectives lead to hebbian/anti-hebbian networks? *Neural computation*, 30(1):84–124, 2018.

[101] J. F. Poulet and C. C. Petersen. Internal brain state regulates membrane potential synchrony in barrel cortex of behaving mice. *Nature*, 454(7206):881–885, 2008.

[102] A. Renart, N. Brunel, and X.-J. Wang. Mean-field theory of irregularly spiking neuronal populations and working memory in recurrent cortical networks. *Computational neuroscience: A comprehensive approach*, pages 431–490, 2004.

[103] A. Renart, J. De La Rocha, P. Bartho, L. Hollender, N. Parga, A. Reyes, and K. D. Harris. The asynchronous state in cortical circuits. *science*, 327(5965):587–590, 2010.

[104] R. A. Rescorla, A. R. Wagner, et al. A theory of pavlovian conditioning: Variations in the effectiveness of reinforcement and nonreinforcement. *Classical conditioning II: Current research and theory*, 2:64–99, 1972.

[105] B. D. Ripley and N. L. Hjort. *Pattern Recognition and Neural Networks*. Cambridge University Press, Jan. 1996.

[106] P. H. Rudebeck, R. C. Saunders, A. T. Prescott, L. S. Chau, and E. A. Murray. Prefrontal mechanisms of behavioral flexibility, emotion regulation and value updating. *Nature Neuroscience*, 16(8):1140–1145, Aug. 2013.

[107] J. Sacramento, R. P. Costa, Y. Bengio, and W. Senn. Dendritic cortical microcircuits approximate the backpropagation algorithm. In *Advances in Neural Information Processing Systems*, pages 8721–8732, 2018.

[108] A. Saez, M. Rigotti, S. Ostojic, S. Fusi, and C. D. Salzman. Abstract Context Representations in Primate Amygdala and Prefrontal Cortex. *Neuron*, 87(4):869–881, Aug. 2015.

[109] M. Sarafyazd and M. Jazayeri. Hierarchical reasoning by neural circuits in the frontal cortex. *Science*, 364(6441):eaav8911, May 2019.

[110] C. Savin, P. Joshi, and J. Triesch. Independent component analysis in spiking neurons. *PLoS Comput Biol*, 6(4):e1000757, 2010.

[111] B. Schölkopf, A. J. Smola, M. D. o. t. M. P. I. f. B. C. i. T. G. P. B. Scholkopf, and F. Bach. *Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond*. MIT Press, 2002.

[112] N. W. Schuck, M. B. Cai, R. C. Wilson, and Y. Niv. Human Orbitofrontal Cortex Represents a Cognitive Map of State Space. *Neuron*, 91(6):1402–1412, Sept. 2016.

[113] W. Schultz, P. Dayan, and P. R. Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[114] W. Schultz, P. Dayan, and P. R. Montague. A Neural Substrate of Prediction and Reward. *Science*, 275(5306):1593–1599, Mar. 1997.

[115] J. K. Seamans, C. C. Lapish, and D. Durstewitz. Comparing the prefrontal cortex of rats and primates: Insights from electrophysiology. *Neurotoxicity Research*, 14(2):249–262, June 2008.

[116] M. N. Shadlen and W. T. Newsome. The variable discharge of cortical neurons: implications for connectivity, computation, and information coding. *The Journal of neuroscience*, 18(10):3870–3896, 1998.

[117] M. N. Shadlen and W. T. Newsome. Neural Basis of a Perceptual Decision in the Parietal Cortex (Area LIP) of the Rhesus Monkey. *Journal of Neurophysiology*, 86(4):1916–1936, Oct. 2001.

[118] K. Shima and J. Tanji. Role for Cingulate Motor Area Cells in Voluntary Movement Selection Based on Reward. *Science*, 282(5392):1335–1338, Nov. 1998.

[119] E. P. Simoncelli and B. A. Olshausen. Natural image statistics and neural representation. *Annual review of neuroscience*, 24(1):1193–1216, 2001.

[120] B. F. Skinner. *The Behavior of Organisms: An Experimental Analysis*. The Behavior of Organisms: An Experimental Analysis. Appleton-Century, Oxford, England, 1938. Pages: 457.

[121] P. Smolensky. On the proper treatment of connectionism. *Behavioral and brain sciences*, 11(1):1–23, 1988.

[122] E. J. Sondik. The optimal control of partially observable markov processes. Technical report, Stanford Univ Calif Stanford Electronics Labs, 1971.

[123] S. Song, K. D. Miller, and L. F. Abbott. Competitive hebbian learning through spike-timing-dependent synaptic plasticity. *Nature neuroscience*, 3(9):919–926, 2000.

[124] M. T. Spaan and N. Vlassis. Perseus: Randomized Point-based Value Iteration for POMDPs. *Journal of Artificial Intelligence Research*, 24:195–220, Aug. 2005.

[125] B. Speelpenning. Compiling fast partial derivatives of functions given by algorithms. Technical report, Illinois Univ., Urbana (USA). Dept. of Computer Science, 1980.

[126] T. A. Stalnaker, N. K. Cooch, and G. Schoenbaum. What the orbitofrontal cortex does not do. *Nature Neuroscience*, 18(5):620–627, May 2015.

[127] C. K. Starkweather, S. J. Gershman, and N. Uchida. The Medial Prefrontal Cortex Shapes Dopamine Reward Prediction Errors under State Uncertainty. *Neuron*, 98(3):616–629.e6, May 2018.

[128] K. Sugiyama, S. Tagawa, and M. Toda. Methods for Visual Understanding of Hierarchical System Structures. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(2):109–125, Feb. 1981.

[129] L. P. Sugrue. Matching Behavior and the Representation of Value in the Parietal Cortex. *Science*, 304(5678):1782–1787, June 2004.

[130] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. A Bradford Book, USA, 2018.

[131] D. G. R. Tervo, M. Proskurin, M. Manakov, M. Kabra, A. Vollmer, K. Branson, and A. Y. Karpova. Behavioral Variability through Stochastic Choice and Its Gating by Anterior Cingulate Cortex. *Cell*, 159(1):21–32, Sept. 2014.

[132] D. Thalmeier, M. Uhlmann, H. J. Kappen, and R.-M. Memmesheimer. Learning universal computations with spikes. *PLoS computational biology*, 12(6):e1004895, 2016.

[133] E. Todorov. General duality between optimal control and estimation. In *2008 47th IEEE Conference on Decision and Control*, pages 4286–4292. IEEE, 2008.

[134] D. J. Tolhurst, J. A. Movshon, and A. Dean. The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision research*, 23(8):775–785, 1983.

[135] R. Urbanczik and W. Senn. Learning by the dendritic prediction of somatic spiking. *Neuron*, 81(3):521–528, 2014.

[136] T. Uustalu and V. Vene. Comonadic notions of computation. *Electronic Notes in Theoretical Computer Science*, 203(5):263–284, 2008.

[137] H. B. M. Uylings and C. G. van Eden. Chapter 3 Qualitative and quantitative comparison of the prefrontal cortex in rat and in primates, including humans. In H. B. M. Uylings, C. G. Van Eden, J. P. C. De Bruin, M. A. Corner, and M. G. P. Feenstra, editors, *Progress in Brain Research*, volume 85 of *The Prefrontal Its Structure, Function and Cortex Pathology*, pages 31–62. Elsevier, Jan. 1991.

[138] C. van Vreeswijk and H. Sompolinsky. Chaos in neuronal networks with balanced excitatory and inhibitory activity. *Science*, 274(5293):1724, 1996.

[139] P. Vertechi, W. Brendel, and C. K. Machens. Unsupervised learning of an efficient short-term memory network. In *Advances in Neural Information Processing Systems*, pages 3653–3661, 2014.

[140] T. Vogels, H. Sprekeler, F. Zenke, C. Clopath, and W. Gerstner. Inhibitory plasticity balances excitation and inhibition in sensory pathways and memory networks. *Science*, 334(6062):1569–1573, 2011.

[141] D. S. Weld and G. Bansal. The challenge of crafting intelligible intelligence. *Communications of the ACM*, 62(6):70–79, 2019.

[142] J. C. Whittington and R. Bogacz. An approximation of the error backpropagation algorithm in a predictive coding network with local hebbian synaptic plasticity. *Neural computation*, 29(5):1229–1262, 2017.

[143] G. N. Wilkinson and C. E. Rogers. Symbolic description of factorial models for analysis of variance. *Applied statistics*, 1973.

[144] S. S. Wilks. The Large-Sample Distribution of the Likelihood Ratio for Testing Composite Hypotheses. *The Annals of Mathematical Statistics*, 9(1):60–62, Mar. 1938.

[145] Z. M. Williams, G. Bush, S. L. Rauch, G. R. Cosgrove, and E. N. Eskandar. Human anterior cingulate neurons and the integration of monetary reward with motor responses. *Nature Neuroscience*, 7(12):1370–1375, Dec. 2004.

[146] R. C. Wilson, Y. K. Takahashi, G. Schoenbaum, and Y. Niv. Orbitofrontal Cortex as a Cognitive Map of Task Space. *Neuron*, 81(2):267–279, Jan. 2014.

[147] A. Wohrer, M. D. Humphries, and C. K. Machens. Population-wide distributions of neural activity during perceptual decision-making. *Progress in neurobiology*, 103:156–193, 2013.

[148] M. Xue, B. V. Atallah, and M. Scanziani. Equalizing excitation-inhibition ratios across visual cortical neurons. *Nature*, 511(7511):596–600, 2014.

[149] P. Yin, J. Fritz, and S. Shamma. Rapid spectrotemporal plasticity in primary auditory cortex during behavior. *J Neurosci*, 34(12):4396–408, 2014.

[150] J. Yu and D. Ferster. Membrane potential synchrony in primary visual cortex during sensory stimulation. *Neuron*, 68(6):1187–1201, 2010.

[151] J. Zylberberg, J. T. Murphy, and M. R. DeWeese. A sparse coding model with synaptically local plasticity and spiking neurons can account for the diverse shapes of v1 simple cell receptive fields. *PLoS Comput Biol*, 7(10):e1002250, 2011.

ITQB-UNL | Av. da República, 2780-157 Oeiras, Portugal
Tel (+351) 214 469 100 | Fax (+351) 214 411 277

www.itqb.unl.pt

Apoio financeiro da FCT e do FSE no âmbito do Quadro Comunitário de Apoio, Bolsa n. PD/BD/105944/2014.